Universidade dos Açores
Departamento de Matemática

Contributos para o estudo da semântica de linguagens com concorrência e mobilidade

Elisabete Maria da Silva Raposo Freire

Ponta Delgada
Novembro 2004

Contributos para o estudo da semântica de linguagens com concorrência e mobilidade

Elisabete Maria da Silva Raposo Freire

Dissertação submetida à Universidade dos Açores para obtenção do grau de Doutor em Informática na especialidade de Teoria da Computação, orientada pelo Professor Luís Fernando Lopes Monteiro, Professor Catedrático do Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

Ponta Delgada
Novembro de 2004

Este trabalho foi parcialmente financiado pela Fundação Para a Ciência e a Tecnologia no âmbito da Bolsa de Investigação SFRH/BD/9047/2002.

Ao Zé, à Andreia e à Catarina

Resumo

Neste trabalho mostra-se como abordagens matematicamente mais simples que as tradicionais podem ser usadas na definição da semântica de linguagens com concorrência e também de linguagens com mobilidade. Completam-se também alguns aspectos da semântica da mobilidade para os quais as propostas actualmente existentes apresentam limitações.

Faz-se a exploração de dois tipos de técnicas para definir as semânticas operacionais e denotacionais de linguagens com concorrência e mobilidade. Por um lado usam-se os conjuntos com famílias de equivalência, um conceito mais simples e manejável, que parece substituir com vantagens ao nível da simplicidade os espaços métricos, usados nas abordagens tradicionais. Por outro lado, completa-se o trabalho com uma abordagem mais recente, baseada na utilização de coálgebras para definir sistemas e tirando partido das facilidades proporcionadas pelos conjuntos nominais na manipulação de nomes.

A avaliação destas técnicas é feita sobre uma linguagem com sincronização restrita ($\mathcal{L}_{syn}$) e sobre uma linguagem com mobilidade (cálculo-π).

Abstract

We show how the semantics of languages involving concurrency and mobility can be defined using approaches that are mathematically simpler then the traditional ones. We also complete some aspects from mobility semantics for which the existent proposals show some limitations.

This work explores two types of techniques to define operational and denotational semantics of languages with concurrency and mobility. On the one hand we use sets with families of equivalence, a simpler and manageable concept that seems to substitute with advantages to the level of simplicity, the metric spaces. On the other hand, we complete the work with a more recent approach, based on the use of coalgebras to define systems and taking advantage on the facilities provided by nominal sets on manipulating names.

The evaluation of these techniques is made on a language with restricted synchronization ($\mathcal{L}_{syn}$) and on a language with mobility (π-calculus).

Agradecimentos

Ao meu orientador, Professor Doutor Luís Monteiro, pelo estímulo dado, pelo acompanhamento atento, pelas sugestões e críticas feitas que foram extremamente úteis para a elaboração desta dissertação. Foi um privilégio tê-lo como orientador.

A todos os colegas e amigos que, directa ou indirectamente, me apoiaram ao longo deste importante percurso da minha vida.

Aos meus pais, foi neles que tudo começou.

Às minhas filhas, Andreia e Catarina, que esperaram pacientemente que eu terminasse este trabalho.

Ao meu marido, pelo estímulo dado e pela colaboração no alívio das responsabilidades familiares.

A realização deste trabalho exigiu, muitas vezes, a abdicação de algum tempo na companhia da minha família mais próxima, sobretudo durante as deslocações ao continente. Agradeço a compreensão com que suportaram estas ausências.

À Universidades dos Açores, em particular ao Departamento de Matemática, pelo apoio logístico e pela dispensa de serviço docente.

À Fundação para a Ciência e a Tecnologia pelo apoio financeiro.

A todos o meu muito obrigada!

Conteúdo

Capítulo 1

Introdução

1.1 Contexto e Contribuições

O objectivo central desta dissertação é contribuir para o constante desenvolvimento do estudo da semântica da concorrência com especial ênfase nas situações que envolvem mobilidade. Apesar de já existirem diversos trabalhos nessas áreas, procuramos instrumentos mais simples de usar, que permitirão aplicações mais acessíveis e a generalização da utilização das técnicas semânticas a um maior número de pessoas e a um maior número de situações. Nesse sentido, exploramos a utilização dos conjuntos com famílias de equivalência, um conceito que matematicamente é muito simples, e que se apresenta suficientemente expressivo para lidar com modelos concorrentes. A semântica da mobilidade ainda é um campo onde há muito a fazer. As propostas actualmente existentes são muito complexas e apresentam limitações várias e critérios que não são contemplados, como é o caso da composicionalidade. A nossa contribuição para a semântica da mobilidade tem como base os conjuntos nominais e as coálgebras. Os conjuntos nominais pela sua grande capacidade de manipular os nomes, um factor fundamental na mobilidade. As coálgebras por se apresentar vantajoso trabalhar com sistemas em vez de conjuntos estruturados. Em resumo, as nossas preocupações dominantes

foram a procura de instrumentos matemáticos simples e a consideração de problemas abordados deficientemente na semântica da mobilidade.

As construções semânticas têm subjacentes conceitos matemáticos bastante complexos. Normalmente, a definição da semântica exige o cálculo de pontos fixos para definições recursivas de domínios semânticos e de denotações nesses domínios de entidades sintácticas. A matemática para isso é complicada e as abordagens tradicionais consistem em utilizar modelos semânticos baseados em certos tipos de conjuntos parcialmente ordenados ou em espaços métricos. No primeiro caso as denotações semânticas são obtidas utilizando o teorema do ponto fixo de Knaster-Tarski, e no segundo o teorema do ponto fixo de Banach. Em ambos os casos a perspectiva mais geral de resolução de equações de domínios baseia-se no trabalho de Smyth e Plotkin, [SP82], desenvolvido originalmente para estruturas ordenadas e adaptado posteriormente às estruturas métricas por America e Rutten em [AR89]. A teoria dos conjuntos com famílias de equivalência, abreviadamente cfe's, foi desenvolvida em [Mon98] com o intuito de simplificar a matemática envolvida nesse processo. Os cfe's constituem uma especialização dos espaços métricos e a respectiva teoria uma adaptação dos conceitos e técnicas dos espaços métricos. Esta simplificação dos espaços métricos mostra-se suficientemente poderosa para as definições semânticas mais comuns.

Como um dos objectivos da dissertação foi simplificar a descrição da semântica das linguagens da concorrência usando cfe's em vez de espaços métricos, usamos para comparação o livro de De Bakker e De Vink [BV96], uma obra onde os autores exploram, utilizando técnicas métricas, 27 modelos matemáticos de linguagens de laboratório. Foi estudado um largo número dessas linguagens modelo, e todas as técnicas de descrição semântica, usadas em [BV96], puderam ser adaptadas aos cfe's com a vantagem de se obterem descrições mais simples sem perda de generalidade. Onde os autores utilizam os espaços métricos, aqui são utilizados os conjuntos com famílias de equivalência, e as demonstrações, que no caso dos espaços métricos são feitas

com base nas distâncias, aqui são feitas por indução, o que permite simplificações significativas em muitas situações. Ilustramos aqui a aplicação destas técnicas, apresentando o estudo efectuado sobre um destes modelos, a linguagem $\mathcal{L}_{syn}$. Trata-se de uma linguagem com uma estrutura simples mas que ilustra bem alguns dos aspectos presentes nas linguagens concorrentes mais comuns como a composição sequencial, a recursividade, a escolha não determinista e a composição paralela com sincronização restrita.

Procurou-se ir para além das linguagens de laboratório de De Bakker e De Vink e aplicar a teoria ao cálculo-π, que do ponto de vista semântico introduz noções relacionadas com a manipulação de nomes que não tinham aparecido até aqui e portanto não tinham sido objecto de estudo. Chegou-se à conclusão de que eram exigidos conceitos completamente novos e as abordagens inspiradas no trabalho de De Bakker e De Vink (escola de Amsterdão da Concorrência) revelaram-se desadequadas e insuficientes. Tornou-se evidente que tínhamos de procurar uma abordagem alternativa, e a escolha acabou por recair na teoria das coálgebras, uma teoria recente que parece bastante promissora no que diz respeito à definição da semântica.

As coálgebras são modelos matemáticos muito gerais, particularmente adequados para modelar sistemas dinâmicos, por isso adaptam-se muito bem à semântica da concorrência e da mobilidade. Além disso, nas técnicas coalgébricas os resultados aplicam-se directamente na categoria dos conjuntos, sem necessidade de considerar estruturas ordenadas ou métricas o que dá uma certa simplicidade à teoria. Esta simplicidade revela-se ainda no facto de não se exigir o cálculo explícito de pontos fixos para as denotações semânticas, em seu lugar utilizam-se definições coindutivas.

Os nomes desempenham um papel fundamental nas questões de mobilidade e em particular no cálculo-π. Na nossa opinião, um dos ponto fracos da maioria dos modelos propostos para o cálculo-π é a forma como estes lidam com os nomes. Nesta dissertação, procurou-se ultrapassar esse problema utilizando a recente teoria dos conjuntos nominais ([Pit01, GP02, Mon03]).

Trata-se de uma teoria simples mas muito promissora e poderosa no que diz respeito às técnicas para lidar com as permutações de nomes.

Como passo preliminar, aplica-se a teoria das coálgebras e dos conjuntos nominais à descrição da semântica de uma linguagem para a qual também se tinham utilizado os cfe's. As vantagens relativas das coálgebras não são evidentes neste caso por se tratar de uma linguagem comparativamente simples. O maior benefício é estabelecer a metodologia a seguir e a refinar no caso do cálculo-π. Procurou-se, em seguida, modelar a semântica do cálculo-π usando coálgebras e conjuntos nominais numa perspectiva inspirada em [Sta96] mas, enquanto Stark usou estruturas ordenadas, no nosso trabalho, usamos as coálgebras e o papel dos nomes é captado pelos conjuntos nominais, em vez de ser por meio de functores. Neste caso as vantagens da utilização das novas técnicas são evidentes.

Com este trabalho, demonstra-se que os cfe's podem ser usados com vantagem sobre os espaços métricos na semântica da concorrência e demonstra-se ainda a grande capacidade de expressão das técnicas coalgébricas, aliadas à dos conjuntos nominais, para estudar sistemas móveis baseados na manipulação de nomes.

1.2 Organização do trabalho

Esta tese é constituída por um capítulo de introdução; um capítulo de conceitos e resultados acessórios; seguido pelo corpo principal do trabalho, formado pelos capítulos 3 e 4; e, por último, um capítulo de considerações finais.

Na introdução descreve-se o contexto em que surgiu e se desenvolveu o trabalho e a organização do mesmo.

No capítulo 2 são expostas as teorias de base sobre as quais se desenvolvem os dois capítulos seguintes. Está dividido em três secções. Na primeira, apresentam-se os cfe's, incluindo resultados de convergência e completude, e

alguns functores, necessários para a futura definição dos domínios semânticos e equações de domínios sobre cfe's. Na segunda secção introduzem-se os conjuntos-Π, os conjuntos nominais e estudam-se alguns functores sobre os conjuntos nominais que irão ser utilizados na definição dos domínios semânticos do cálculo-π. A terceira e última secção contém os conceitos básicos das coálgebras e alguns resultados de caracterização da bissimilaridade e equivalência semântica.

No decorrer do capítulo 3, faz-se o estudo da semântica da linguagem $\mathcal{L}_{syn}$. Este está dividido em quatro secções. Na primeira é apresentada $\mathcal{L}_{syn}$. Na segunda é definida a sua semântica operacional e a sua semântica denotacional utilizando os cfe's e mostra-se a equivalência entre as duas. Na terceira secção, é feito um estudo semelhante mas para as semânticas definidas utilizando coálgebras. Na quarta secção, termina-se fazendo um estudo comparativo dos resultados obtidos pelos dois métodos.

O capítulo 4 é dedicado ao estudo da semântica do cálculo-π e está dividido em seis secções. Na primeira é apresentado o cálculo-π e algumas definições básicas. Na segunda secção é definido o domínio semântico que iremos utilizar nas secções seguintes. Na terceira e quarta secções, são utilizadas técnicas coalgébricas para definir, respectivamente, uma semântica operacional e uma semântica denotacional para o cálculo-π. Na quinta secção prova-se a equivalência entre a semântica operacional e a semântica denotacional, no caso fechado. Terminamos mostrando, na sexta secção, que o núcleo de equivalência das semânticas coincide com a bissimilaridade forte, no caso fechado, e com a congruência forte no caso aberto.

No quinto e último capítulo são apresentadas algumas considerações finais sobre os objectivos atingidos e os possíveis desenvolvimentos que se poderão seguir.

Capítulo 2

Preliminares

Este capítulo fornece conceitos de base necessários para o resto do trabalho. Está dividido em três partes. A primeira é dedicada aos conjuntos com famílias de equivalência, denotados por cfe´s. Aí, para além das definições básicas, apresentam-se resultados de convergência e completude e estudam-se alguns functores, necessários para a futura definição dos domínios semânticos. Na segunda secção introduzem-se os conjuntos-Π, os conjuntos nominais e estudam-se alguns functores sobre os conjuntos nominais que irão ser utilizados, no capítulo 4, na definição dos domínios semânticos do cálculo-π. A terceira e última secção contém os conceitos básicos das coálgebras e alguns resultados de caracterização da bissimilaridade e equivalência semântica.

2.1 Conjuntos com famílias de equivalência

Nesta secção são apresentadas algumas definições e resultados baseados na teoria apresentada em [Mon98], bem como outros resultados no âmbito dos conjuntos com famílias de equivalências que serão necessários no decorrer do capítulo 3.

2.1.1 Definições básicas

Começamos por introduzir as noções elementares da teoria dos conjuntos com famílias de equivalência (cfe's).

Definição 2.1.1 (Conjunto com uma família de equivalências) Um *conjunto com uma família de equivalências*, abreviadamente cfe, é um conjunto S junto com uma família $(\equiv_n)_{n\geq 0}$ de relações de equivalência em S tais que

- $\equiv_0$ é a relação $S \times S$;
- $\equiv_{n+1} \subseteq \equiv_n$ para todo o $n \geq 0$.

Denotamos a intersecção das $\equiv_n$ por $\equiv_\omega$. A família de equivalências é *separadora*, e S é um cfe *separado*, se $\equiv_\omega$ é a relação de identidade em S. □

Para aliviar a escrita iremos denotar um cfe $\langle S, (\equiv_n)_{n\geq 0}\rangle$ apenas pelo conjunto S que lhe está subjacente, deixando implícita a relação $\equiv_n$. Quando houver vários cfe's sob consideração, poderemos ter de usar a notação $\equiv_n^S$ para explicitar qual o cfe a que nos queremos referir.

Deste modo, dados $s, t \in S$, tem-se $s \equiv_\omega t$ se e só se $s \equiv_n t$ para todo $n \geq 0$. Num cfe separado $s \equiv_\omega t$ implica $s = t$ e, portanto, sempre que $s \neq t$ é possível encontrar um $n \geq 0$ tal que $s \not\equiv_n t$. Neste caso, existe um maior k tal que $s \equiv_k t$.

Dada a relação $\equiv_\alpha$, onde α é um número natural n ou ω, a classe de equivalência de s é $[s]_\alpha = \{t : s \equiv_\alpha t\}$. Pelos axiomas que definem os cfe vem que $[s]_0 = S$ e $[s]_{n+1} \subseteq [s]_n$ para todo o s e n. Tem-se ainda que $[s]_\omega = \cap_n [s]_n$, e S é separado se e só se $[s]_\omega = \{s\}$ para todo o s.

Vamos de seguida apresentar alguns exemplos de cfe's.

Qualquer conjunto S pode ser visto como um cfe tomando, para todo o $n > 0$, $\equiv_n$ como a relação identidade em S. Esse tipo de cfe será designado por *discreto*. Deste modo, a categoria Set pode ser encarada como uma categoria de cfe's. Naturalmente que os cfe's discretos são separados.

Dados os naturais m, n, k, defina-se $m \equiv_n k$ se e só se $m = k$ ou $n \leq min\{m, k\}$. As classes de equivalência de $\equiv_n$ no conjunto dos naturais $\omega = \{0, 1, ...\}$ são então $\{m\}$ para $m < n$ e $\{m : m \geq n\}$. É fácil verificar que com esta definição ω é um cfe separado. Esta é a estrutura usual ao considerarmos ω como um cfe.

Outro exemplo importante é $A^\infty = A^* \cup A^\omega$, o conjunto das sequências finitas (A^*) e infinitas (A^ω) sobre um alfabeto A, com a família de relações definida do modo que se segue. Dadas as sequências u e v, $u \equiv_n v$ se e só se $u = v$ ou u e v têm comprimento maior ou igual a n e têm o mesmo prefixo de comprimento n. Com esta família de relações, A^∞ constitui um cfe separado.

Podemos encarar um sistema de transições $\langle S, A, \rightarrow \rangle$, sobre um alfabeto A, como um conjunto S junto com a relação $\rightarrow \subseteq S \times A \times S$. Para termos um cfe basta considerarmos $s \equiv_n t$ se e só se s e t têm os mesmos traços de comprimento menor ou igual a n, isto é, para todo $0 < k \leq n$ e $a_1, ..., a_k \in A$, existe uma sequência de transições $s \xrightarrow{a_1} ... \xrightarrow{a_k} s'$ se e só se existe uma sequência semelhante $t \xrightarrow{a_1} ... \xrightarrow{a_k} t'$. Esta é a perspectiva que vai ter mais interesse no nosso estudo e que vamos ver mais em pormenor quando definirmos a semântica de uma linguagem a partir de um sistema de transições sobre cfe's.

Definição 2.1.2 (Conservadora, aproximante) Uma função $f : S \rightarrow T$ entre cfe's é *conservadora* se $s \equiv_n t$ implica $f(s) \equiv_n f(t)$ para todo o $s, t \in S$ e $n \geq 0$. A função é *aproximante* se $f(s) \equiv_{n+1} f(t)$ sempre que $s \equiv_n t$, para $n \geq 0$. □

Como as funções identidade são conservadoras e a composição de funções conservadoras é ainda conservadora, é imediato que os cfe's com as funções conservadoras constituem uma categoria que será denotada por Cfe. A subcategoria dos cfe's separados é denotada por SCfe. As funções aproximantes são conservadoras – note-se que $\equiv_{n+1} \subseteq \equiv_n$ – mas não definem uma subcategoria de Cfe uma vez que as funções identidade não são, em geral, aproximantes.

Lema 2.1.3 *Se $f : S \to T$ e $g : T \to U$ são funções conservadoras e uma de entre f e g é aproximante, a composição $g \circ f$ é aproximante.*

Demonstração Suponhamos que f é aproximante. Então se $s \equiv_n t$ em S tem-se $f(s) \equiv_{n+1} f(t)$ em T e como g é conservadora $g(f(s)) \equiv_{n+1} g(f(t))$ em U. Se for g a função aproximante a demonstração é similar. □

2.1.2 Operações sobre cfe's

Introduzimos aqui algumas operações e functores sobre cfe's que serão úteis para a definição do functor que irá servir para o cálculo dos domínios semânticos da linguagem que vamos estudar no capítulo 3.

A primeira operação atenua as diferenças entre os elementos de um cfe ao substituir cada $\equiv_n$ por uma outra relação $\equiv_n^\circ$ que identifica mais elementos e por isso é designada por atenuação.

Definição 2.1.4 (Atenuação) Seja S um cfe. A *atenuação* de S, denotada por S°, é o cfe constituído pelo mesmo conjunto mas com as relações de equivalência $\equiv_n^\circ$ definidas por $\equiv_0^\circ = \equiv_0$ e $\equiv_{n+1}^\circ = \equiv_n$ para todo o $n \geq 0$. □

Pela definição é imediato que S° é um cfe separado se S for separado e que a função identidade em S é aproximante quando tomada como uma função de S em S°. Mostra-se facilmente por indução, que uma função conservadora (aproximante) $f : S \to T$ é também conservadora (aproximante) quando encarada como função de S° em T°. Vejamos o caso em que f é aproximante, o caso em que é conservadora é análogo. Para $n = 0$, como $\equiv_1^\circ = \equiv_0$, se $s \equiv_0^\circ t$ então $f(s) \equiv_1^\circ f(t)$. Para $n + 1$, se $s \equiv_{n+1}^\circ t$ é porque $s \equiv_n t$ e portanto $f(s) \equiv_{n+1} f(t)$, donde $f(s) \equiv_{n+2}^\circ f(t)$. Estes resultados mostram que a atenuação é um functor em Cfe e em SCfe, usualmente denotado por Id°, que aplica cada cfe S em S° e cada função conservadora f em $f^\circ = f$.

Definição 2.1.5 (Produto) Sejam S e T cfe's. O *produto* $S \times T$ é o produto cartesiano usual dos conjuntos S e T com as equivalências definidas por

$$(s,t) \equiv_n (s',t') \text{ se e só se } s \equiv_n s' \text{ e } t \equiv_n t'.$$

□

O próximo resultado mostra que $S \times T$ é um produto na categorias Cfe e SCfe.

Proposição 2.1.6 *Se S e T são cfe's, $S \times T$ é um cfe, que é separado se S e T o forem. As projecções $\pi_1 : S \times T \to S$ e $\pi_2 : S \times T \to T$, onde $\pi_1(s,t) = s$ e $\pi_2(s,t) = t$, são conservadoras. Dado outro cfe R e funções conservadoras (aproximantes) $f : R \to S$ e $g : R \to T$, a única função $\langle f, g\rangle : R \to S \times T$ tal que $\langle f, g\rangle(r) = (f(r), g(r))$ é conservadora (aproximante).*

Demonstração Todos os resultados são fáceis de verificar, vamos ver apenas a última afirmação para o caso das funções serem conservadoras. Sejam $s, t \in R$ com $s \equiv_n t$. Como f e g são conservadoras tem-se $f(s) \equiv_n f(t)$ e $g(s) \equiv_n g(t)$, logo $(f(s), g(s)) \equiv_n (f(t), g(t))$ que é o mesmo que $\langle f, g\rangle(s) \equiv_n \langle f, g\rangle(t)$. □

O produto pode ser estendido a pares de funções, definindo um functor binário em Cfe ou SCfe. Dadas $f : S \to S'$ e $g : T \to T'$, o respectivo produto $f \times g : S \times T \to S' \times T'$ é definido por $f \times g = (f \circ \pi_1, g \circ \pi_2)$, ou seja $f \times g(s,t) = (f(s), g(t))$.

Definição 2.1.7 (Soma) Sejam S e T cfe's. A *soma* $S + T$ é o conjunto $\{1\} \times S \cup \{2\} \times T$ com as equivalências definidas por

$$(a,s) \equiv_0 (b,t), \text{ para todo o } (a,s),(b,t) \in S + T;$$
$$\text{para } n > 0, (a,s) \equiv_n (b,t) \text{ se e só se } a = b \text{ e } s \equiv_n t.$$

□

Vamos agora mostrar que definida deste modo $S+T$ é um coproduto em Cfe e em SCfe.

Proposição 2.1.8 *Se S e T são cfe's, $S+T$ é um cfe, que é separado se S e T o forem. As injecções $i_1 : S \to S+T$ e $i_2 : T \to S+T$, onde $i_1(s) = (1,s)$ e $i_2(t) = (2,t)$, são conservadoras. Dado outro cfe U e funções conservadoras $f : S \to U$ e $g : T \to U$, a única função $[f,g] : S+T \to U$ tal que $[f,g](1,s) = f(s)$ e $[f,g](2,t) = g(t)$ é conservadora.*

Demonstração Todos os resultados são fáceis de verificar. □

Tal como o produto, a soma pode ser estendida facilmente a pares de funções, definindo um functor binário em Cfe ou SCfe. Dadas $f : S' \to S$ e $g : T' \to T$, define-se $f+g : S'+T' \to S+T$ por $f+g = [i_1 \circ f, i_2 \circ g]$, isto é $f(s,t) = (i_1 \circ f(s), i_2 \circ g(t))$.

Definição 2.1.9 (Espaço de funções) Dados os cfe's S e T, o *espaço de funções de S em T*, denotado por $[S \to T]$, é o conjunto de todas as funções conservadoras de S em T, onde $f \equiv_n g$ se e só se $f(s) \equiv_n g(s)$ para todo o $s \in S$. □

Se S for um conjunto visto como um cfe, toda a função $f : S \to T$ é conservadora e, neste caso, $[S \to T]$ é simplesmente o conjunto T^S de todas as funções de S em T.

Proposição 2.1.10 *Se S e T são cfe's, $[S \to T]$ é um cfe, que é separado se T for separado.*

Demonstração Claramente $[S \to T]$ é um cfe. Para concluir que $[S \to T]$ é separado se T o é, vamos tomar f, $f' \in [S \to T]$ com $f \equiv_n f'$ para todo o $n \geq 0$. Então, para todo o $s \in S$, $f(s) \equiv_n f'(s)$ para todo o $n \geq 0$, portanto, como T é separado, $f(s) = f'(s)$. □

Definição 2.1.11 (Domínios potência) Seja S um cfe e X e Y subconjuntos de S. Definam-se as seguintes equivalências

$X \equiv_0 Y$;
para $n > 0, X \equiv_n Y$ se para todo o $x \in X$ existe um $y \in Y$ tal que $x \equiv_n y$, e para todo o $y \in Y$ existe um $x \in X$ tal que $x \equiv_n y$.

Denotamos por $\mathcal{P}(S)$ o conjunto potência de S, isto é, o conjunto de todos os subconjuntos de S, e por $\mathcal{P}_{nv}(S)$ o conjunto de todos os subconjuntos não vazios de S. □

Note-se que se $X \subseteq Y$ para garantir que $X \equiv_n Y$, para $n > 0$, precisamos apenas de provar que para todo o $y \in Y$ existe um $x \in X$ tal que $x \equiv_n y$.

Proposição 2.1.12 *Se S é um cfe então $\mathcal{P}(S)$ e $\mathcal{P}_{nv}(S)$ são cfe's. Se $f : S \to T$ é conservadora, a função $\mathcal{P}(f) : \mathcal{P}(S) \to \mathcal{P}(T)$ definida por $\mathcal{P}(f)(X) = \{f(s) : s \in X\}$ e a sua restrição $\mathcal{P}_{nv}(f) : \mathcal{P}_{nv}(S) \to \mathcal{P}_{nv}(T)$ são conservadoras. Além disso, se f é aproximante então $\mathcal{P}_{nv}(f)$ também é aproximante.*

Demonstração É imediato que $\mathcal{P}(S)$ e $\mathcal{P}_{nv}(S)$ são cfe's. Para provar que $\mathcal{P}(f)$ é conservadora, suponha-se que $X \equiv_n Y$ em $\mathcal{P}(S)$. Se X e Y são ambos vazios, então são iguais e a conclusão é imediata. Se um dos conjuntos é vazio e o outro não, então tem de ser $n = 0$ e, novamente, a conclusão é imediata. Vamos então mostrar que $\mathcal{P}(f)(X) \equiv_n \mathcal{P}(f)(Y)$ no caso em que nenhum dos conjuntos é vazio. Tomemos um elemento em $\mathcal{P}(f)(X)$, que tem necessariamente a forma $f(x)$ para algum $x \in X$, como $X \equiv_n Y$, existe um $y \in Y$ tal que $x \equiv_n y$. Mas f é conservadora, logo $f(x) \equiv_n f(y)$. Do mesmo modo, dado um elemento em $\mathcal{P}(f)(Y)$ podemos encontrar um elemento em $\mathcal{P}(f)(X)$ que está na relação $\equiv_n$ com o primeiro. A demonstração que $\mathcal{P}_{nv}(f)$ é aproximante se f o for, é análoga a esta última parte. □

De notar que em geral $\mathcal{P}(S)$ e $\mathcal{P}_{nv}(S)$ não são separados mesmo que S o seja. Por esta razão $\mathcal{P}(S)$ e $\mathcal{P}_{nv}(s)$ são functores em Cfe, mas a sua restrição a SCfe não é um functor.

Proposição 2.1.13 *Sejam $f, g : S \to T$ funções conservadoras tais que $f \equiv_n g$ em $[S \to T]$. Se $k : R \to S$ e $h : T \to U$ são conservadoras então $f \circ k \equiv_n g \circ k$ em $[R \to T]$ e $h \circ f \equiv_n h \circ g$ em $[S \to U]$. Em particular, a operação composição definida de $[T \to U] \times [S \to T]$ em $[S \to U]$, que a cada (f, g) faz corresponder $g \circ f$, é conservadora.*

Demonstração Para todo o $r \in R$, $f(k(r)) \equiv_n g(k(r))$, logo $f \circ k \equiv_n g \circ k$. Para todo o $s \in S$, $f(s) \equiv_n g(s)$, logo $h(f(s)) \equiv_n h(g(s))$ e portanto $h \circ f \equiv_n h \circ g$. Em relação à última afirmação, se $(g, f) \equiv_n (g', f')$ em $[T \to U] \times [S \to T]$ então $g \equiv_n g'$ e $f \equiv_n f'$ e logo, pela primeira parte da proposição, $g \circ f \equiv_n g' \circ f \equiv_n g' \circ f'$. □

2.1.3 Convergência e completude

Nesta subsecção vamos apresentar alguns resultados de convergência e completude para cfe's, bem como um teorema de ponto fixo para cfe's completos e separados, que é análogo ao teorema do ponto fixo de Banach para os espaços métricos. Muitos dos resultados, cuja demonstração pode ser consultada em [Mon98], não serão aqui demonstrados.

Definição 2.1.14 (Limite) Seja S um cfe. Uma sucessão $(s_n)_{n\geq 0}$ de elementos de S *converge* para $s \in S$, ou tem *limite* s, se $s_n \equiv_n s$ para todo o $n \geq 0$. Neste caso a sucessão diz-se *convergente*. □

Como o conjunto índice das sucessões é sempre o conjunto dos naturais vamos muitas vezes abreviar $(s_n)_{n\geq 0}$ para simplesmente (s_n).

Se (s_n) converge para s, cada s_n fornece alguma informação sobre o limite s, em particular as classes de equivalência $[s]_n$ e $[s_n]_n$ são iguais. Deste modo, a sequência não crescente $[s_0]_0 \supseteq [s_1]_1 \supseteq ...$ tem como intersecção $[s]_\omega$. Reciprocamente, todo o s' na intersecção está em todas as classes $[s_n]_n$ e, portanto é um limite de (s_n). Isto permite-nos concluir que num cfe separado o limite de uma sucessão, quando existe, é único. Neste caso é usual denotar o limite por $\lim s_n$. As funções conservadoras podem ser caracterizadas como as funções que preservam os limites das sucessões convergentes, e a convergência interage com as construções apresentadas da forma esperada, conforme mostram as duas proposições que se seguem.

Proposição 2.1.15 *Uma função $f : S \to T$ entre cfe's é conservadora se e só se aplica sucessões convergentes de S em sucessões convergentes de T, preservando os limites.* □

Proposição 2.1.16 *Sejam S, T e U cfe's.*

(i) A sucessão $(s_n, t_n)_n$ em $S \times T$ converge para (s, t) se e só se (s_n) converge para s e (t_n) converge para t.

(ii) Se uma sucessão (f_n) em $[S \to T]$ converge para f então $(f_n(s))_n$ converge para $f(s)$ para todo o $s \in S$.

(iii) Reciprocamente, se $f : S \to T$ é tal que $(f_n(s))_n$ converge para $f(s)$para todo o $s \in S$ então f é conservadora e (f_n) converge para f.

(iv) Se (f_n) converge para f em $[S \to T]$ e (g_n) converge para g em $[T \to U]$ então $(g_n \circ f_n)_n$ converge para $g \circ f$ em $[S \to U]$.

□

Definição 2.1.17 (Sucessão regular, cfe completo) Uma sucessão (s_n) é *regular* se $s_n \equiv_n s_{n+1}$ para todo o $n \geq 0$. O cfe S é *completo* se toda a sucessão regular de S tem limite em S. A subcategoria plena de Cfe dos cfe's completos e separados é denotada por CCfe. □

A condição que define uma sucessão regular é equivalente a $[s_n]_n \supseteq [s_{n+1}]_{n+1}$ para todo o $n \geq 0$, ou seja, (s_n) dá origem a uma sequência $[s_0]_0 \supseteq [s_1]_1 \supseteq ....$ Assim, toda a sequência convergente é regular, mas o inverso nem sempre é verdade pois podemos ter $\bigcap_n [s_n] = \emptyset$.

Outra forma de caracterizar os cfe's completos é pela propriedade que toda a sequência $C_0 \supseteq C_1 \supseteq ...$, onde cada C_n é uma classe de equivalência $\equiv_n$ tem intersecção não vazia.

A noção de cfe completo permite-nos estabelecer o resultado que se segue, que é um caso especial do teorema do ponto fixo de Banach, [Ban22],para os espaços métricos.

Teorema 2.1.18 ***(Banach)*** *Seja S um cfe completo e separado. Uma função aproximante $f : S \to S$ tem um único ponto fixo.*

Demonstração Como f aplica classes $\equiv_n$ em classes $\equiv_{n+1}$, tem-se uma sequência $C_0 \supseteq C_1 \supseteq ...$ onde cada C_n é uma classe de equivalência $\equiv_n$ e $f(C_n) \subseteq C_{n+1}$. Todo o ponto fixo de f está em $\cap_n C_n$. Reciprocamente, se $s \in \cap_n C_n$ então $f(s) \in \cap_n C_n$. Como num cfe completo e separado a intersecção $\cap_n C_n$ é um conjunto singular, $f(s) = s$ e é o único ponto fixo de f. □

A demonstração também se poderia fazer seguindo os passos da prova do teorema de Banach para os espaços métricos.

Naturalmente que qualquer conjunto S quando visto como um cfe é completo e separado.

Os functores apresentados, quando aplicados a cfe's completos, permitem construir outros cfe's que ainda gozam da mesma propriedade, isto é, se S

e T são cfe's completos então S°, $S \times T$, $S + T$, $[S \to T]$, $\mathcal{P}(S)$ e $\mathcal{P}_{nv}(S)$ também o são.

2.1.4 Conjuntos potência

Dado um cfe S, já vimos que $\mathcal{P}(S)$ nem sempre é separado. Vamos então estudar os conjuntos potência $\mathcal{P}_{fe}(S)$ e $\mathcal{P}_{fenv}(S)$ dos subconjuntos fechados e dos subconjuntos fechados e não vazios de S, respectivamente. Veremos ainda o caso das potências $\mathcal{P}_{co}(S)$ e $\mathcal{P}_{conv}(S)$ dos subconjuntos compactos e compactos não vazios de S, respectivamente.

Definição 2.1.19 (Conjunto fechado, fecho) Um subconjunto X de um cfe S é *fechado* se $X \equiv_{\omega} Y$ implica $Y \subseteq X$ para todo o $Y \subseteq S$. O *fecho* de X, denotado por X^c, é a união de todos os $Y \subseteq S$ tais que $X \equiv_{\omega} Y$. □

Estes conceitos também podem ser caracterizados da forma convencional, isto é, seja S um cfe e $X \subseteq S$. O fecho X^c é, simultaneamente, o maior subconjunto de S tal que $X \equiv_{\omega} X^c$ e o menor dos subconjuntos fechados que contêm X. Além disso, X^c é o conjunto dos limites das sucessões convergentes de elementos de X. Um conjunto X é fechado se e só se $X^c \subseteq X$.

Definição 2.1.20 (Potências dos conjuntos fechados) Denotam-se por $\mathcal{P}_{fe}(S)$ e $\mathcal{P}_{fenv}(S)$, respectivamente, o conjunto de todos os subconjuntos fechados e o conjunto de todos os subconjuntos fechados não vazios de S. □

Estes conjuntos constituem cfe's com as equivalências $\equiv_n$ definidas como as restrições das correspondentes equivalências em $\mathcal{P}(S)$. Deste modo, $\mathcal{P}_{fe}(S)$ e $\mathcal{P}_{fenv}(S)$ são separados e se S é completo são também completos.

Definição 2.1.21 (Conjunto compacto) Seja S um cfe completo e separado e $X \subseteq S$. Diz-se que X é *compacto* se satisfaz alguma das seguintes condições, que são equivalentes entre si:

(i) X é fechado e toda a equivalência $\equiv_n$ é finitária em X;

(ii) existe uma sucessão (X_n) de subconjuntos finitos de S que converge em $\mathcal{P}_{fe}(S)$ para X;

(iii) toda a sucessão (x_n) em X tem uma subsucessão convergente $(x_{i_n})_n$ com limite em X.

O conjunto dos subconjuntos compactos e o conjunto dos subconjuntos compactos não vazios denotam-se por $\mathcal{P}_{co}(S)$ e $\mathcal{P}_{conv}(S)$, respectivamente. □

Lema 2.1.22 *Seja S um* cfe *completo e separado. Se (X_n) é uma sucessão regular de subconjuntos compactos de S, o seu limite em $\mathcal{P}_{fe}(S)$ é compacto.* □

Tem-se ainda que se S e T são cfe's completos então $\mathcal{P}_{co}(S)$ e $\mathcal{P}_{conv}(S)$ são completos e se $f : S \to T$ é conservadora e $X \subseteq S$ é compacto então $f(X)$ é compacto. Isto permite-nos concluir que $\mathcal{P}_{co}(S)$ e $\mathcal{P}_{conv}(S)$ são endofunctores em CCfe.

2.1.5 Equações de Domínios

Nesta secção vamos procurar soluções para equações de domínios sobre cfe's, isto é, para equações que definem cfe's por recursão através de isomorfismos do tipo $S \cong F(S)$ onde S é um cfe e F um functor sobre cfe's.

Um functor $F : \mathsf{SCfe} \to \mathsf{SCfe}$ aplica uma função $f : S \to T$ em $F(f) : F(S) \to F(T)$. Para cada par S, T, define-se uma aplicação

$$F_{S,T} : [S \to T] \to [F(S) \to F(T)]$$

em que

$$F_{S,T}(F) = F(f).$$

Definição 2.1.23 (Functores localmente conservadores e localmente aproximantes) Um functor $F : \mathsf{SCfe} \to \mathsf{SCfe}$ é localmente conservador (respectivamente, localmente aproximante) se $F_{S,T}$ for conservadora (respectivamente, aproximante) para todo o par S, T. □

Todos os functores apresentados são localmente conservadores. O functor constante e o functor atenuação são ainda localmente aproximantes. Se um functor F for localmente conservador, o functor $F \circ Id^\circ$ é localmente aproximante, onde $(F \circ Id^\circ)(S) = F(Id^\circ(S)) = F(S^\circ)$. Também se denota $F \circ Id^\circ$ simplesmente por F°.

Definição 2.1.24 (Cadeia de cfe's) Uma *cadeia* de cfe's é uma sucessão $(S_n, \beta_n)_n$ de cfe's e funções conservadoras, tais que

$$S_0 \xleftarrow{\beta_0} S_1 \xleftarrow{\beta_1} \dots S_n \xleftarrow{\beta_n} S_{n+1} \dots$$

Um *cone* $\sigma : S \to (S_n, \beta_n)_n$ é um cfe S junto com uma sucessão $\sigma = (\sigma_n)_n$ de morfismos $\sigma_n : S \to S_n$ tais que, para todo o $n \geq 0$,

$$\sigma_n = \beta_n \circ \sigma_{n+1}.$$

□

Definição 2.1.25 (Limite de uma cadeia de cfe's) O limite de uma cadeia regular de cfe's $(S_n, \beta_n)_n$ é o cone $\sigma : S \to (S_n, \beta_n)_n$ onde S é um cfe definido por

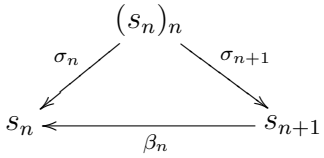
$$S = \{(s_n)_n : \text{ para todo o } n, s_n \in S_n \text{ e } s_n = \beta_n(s_{n+1})\}$$

com as equivalências

$$(s_m)_m \equiv_n (s_k)_k \text{ se e só se } s_l \equiv_n t_l, \text{ para todo o } l,$$

e para todo o n, $\sigma_n : S \to S_n$ é definido por $\sigma_n((s_n)_n) = s_n$. □

Naturalmente que, nestas condições, $\sigma_n = \beta_n \circ \sigma_{n+1}$, isto é, o seguinte diagrama comuta



O resultado que se segue está demonstrado em [Mon98] numa perspectiva mais geral do que a que temos vindo a considerar, mas que excede as nossas necessidades; aqui limitamo-nos a apresentar a construção do ponto fixo sem demonstrar que ele é de facto um ponto fixo e único. Para enunciar o teorema necessitamos de uma definição.

Definição 2.1.26 (Functor não trivial) Um endofunctor F diz-se *não trivial* se $F(S) \neq \emptyset$ sempre que $S \neq \emptyset$. □

Um functor não trivial é também designado por functor não identicamente nulo.

Teorema 2.1.27 *Seja F um endofunctor aproximante e não trivial em* CCfe*, então existe um único S, a menos de isomorfismo, tal que $S \cong F(S)$.*

Demonstração Defina-se S_0 como um cfe completo e separado com um único elemento e $S_{n+1} = F(S_n)$, para todo o $n \geq 0$. Seja $\beta_0 : S_1 \to S_0$ a única função possível e $\beta_{n+1} = F(\beta_n)$. Considere-se S o limite da cadeia $(S_n, \beta_n)_n$, nestas condições, pelo teorema mencionado, tem-se $S \cong F(S)$ único ponto fixo de F. □

Um functor para a semântica de $\mathcal{L}_{syn}$

Na definição da semântica de $\mathcal{L}_{syn}$ iremos considerar, na categoria CCfe, o endofunctor definido por

$$F(Q) = \{p_\varepsilon\} + \mathcal{P}_{co}(A \times Q^\circ)$$

onde A representa um conjunto de acções que poderá incluir ou não as acções de sincronização, conforme se trate da semântica denotacional ou operacional, respectivamente. Como o functor $F'(Q) = \{p_\varepsilon\} + \mathcal{P}_{co}(A \times Q)$ é conservador a utilização do operador de atenuação torna F aproximante o que garante a existência de uma solução única para a equação de domínios $S \cong F(S)$.

De acordo com a secção anterior, define-se $S_0 = \{\emptyset\}$, $S_{n+1} = \{p_\varepsilon\} + \mathcal{P}_{co}(A \times S_n)$, e os morfismos β vêm:

$\beta_0 : S_1 \to S_0$ é definida por $\beta_0(y) = \emptyset, \forall y \in S_1$;

$\beta_{n+1} : S_{n+2} \to S_{n+1}$ é definida por

$$\beta_{n+1}(y) = \begin{cases} p_\varepsilon & \text{se } y = p_\varepsilon \\ \{(b, \beta_n(x)) : (b, x) \in y\} & \text{caso contrário.} \end{cases}$$

A solução da equação $S \cong F(S)$ é então o cfe limite da cadeia, dado por

$$S = \{(s_n)_n : \text{ para todo o } n, s_n \in S_n \text{ e } s_n = \beta_n(s_{n+1})\}.$$

2.2 Conjuntos Nominais

A essência do cálculo-π é a manipulação de nomes, pelo que os modelos semânticos têm de captar essa capacidade. Por isso vamos utilizar a teoria sobre conjuntos nominais recentemente desenvolvida em [GP99], actualmente com uma edição revista e extendida [GP02], e em [Mon03].

Nesta secção vamos introduzir os conjuntos nominais que constituem uma categoria na qual será definido o domínio semântico do cálculo-π. Os conjuntos nominais são conjuntos-Π com suporte finito. Por sua vez os conjuntos--Π são conjuntos onde está definida uma acção de permutação. Vamos ver algumas operações sobre estes conjuntos, como são os morfismos para os conjuntos-Π e definir o functor que será utilizado, no capítulo 4, para definir a semântica do cálculo-π.

2.2.1 Acções de permutação e conjuntos-Π

Aqui introduzimos as permutações, as acções e os conjuntos-Π bem como algumas noções básicas associadas a estes conceitos.

Definição 2.2.1 (Permutação) Dado um conjunto infinito enumerável de nomes $\mathcal{N}$, uma *permutação* sobre $\mathcal{N}$ é uma bijecção $\sigma : \mathcal{N} \to \mathcal{N}$ tal que $\sigma(x) \neq x$ apenas para um número finito de nomes. O conjunto finito $Nuc(\sigma) = \{x \in \mathcal{N} : \sigma(x) \neq x\}$ diz-se o *núcleo de* σ. A permutação com núcleo vazio designa-se por permutação identidade e representa-se por $1_{\mathcal{N}}$ ou simplesmente 1. O conjunto de todas as permutações, que se denota por Π, é um grupo com a operação de composição de funções. □

Dado X subconjunto de $\mathcal{N}$, o conjunto de todas as permutações que são a identidade em X é um subgrupo de Π e representa-se por Π_X. Uma permutação σ é a identidade em $\mathcal{N} - Nuc(\sigma)$ e portanto a sua restrição a $Nuc(\sigma)$ é ainda uma bijecção.

Para facilidade de leitura e escrita, iremos denotar frequentemente $\sigma(x)$ por x^σ. Se $x^\sigma \neq x$ dizemos que σ *renomeia* x, caso contrário dizemos que σ *fixa* x.

Se x e z são nomes distintos, a permutação que aplica x em z e z em x e fixa todos os outros nomes diz-se uma *transposição* e é usualmente representada por (xz) ou (zx). Deste modo, tem-se $x^{(xz)} = z$, $z^{(xz)} = x$ e $w^{(xz)} = w$ para todo o w diferente de x e z. É imediato que $(xz) \circ (xz) = 1$, onde 1 representa a identidade, e portanto $(xz)^{-1} = (xz)$. De modo mais geral, uma *permutação cíclica* ou simplesmente *ciclo*, representada por $(x_1 x_2 \cdots x_n)$ com $n \geq 2$ e $x_1, x_2, \ldots, x_n$ nomes distintos entre si, aplica x_i em x_{i+1} se $1 \leq i < n$ e aplica x_n em x_1, mantendo fixos todos os outro nomes. Quando $n = 2$ é conveniente permitirmos que os nomes sejam iguais e nesse caso (xx) corresponde à permutação identidade.

Como $(x_1 x_2 \cdots x_n) = (x_1 x_2) \circ (x_2 x_3) \circ \cdots \circ (x_{n-1} x_n)$, um ciclo é uma composição de transposições. Toda a permutação é uma composição de ci-

clos, logo é uma composição de transposições. De facto, se x é um nome renomeado por σ, existe um $n \geq 1$ tal que $\sigma^{n+1}(x) = x$, mas, como σ renomeia apenas um número finito de nomes, temos um ciclo $(x_0 x_1 \cdots x_n)$ com $x_i = \sigma^i(x)$ para $0 \leq i \leq n$, e portanto os nomes renomeados por σ dão origem a vários ciclos disjuntos, que compostos por qualquer ordem dão origem a σ. Uma permutação pode ser vista como um caso particular das substituições, a permutação (xz) corresponde à substituição $\{x/z, z/x\}$.

Definição 2.2.2 (Acção, Conjunto-Π) Uma *acção* de Π sobre um conjunto Q é uma função de $\Pi \times Q$ em Q que se escreve $\sigma.q$ e satisfaz as seguintes propriedades:

$$1 \cdot q = q \ (\textit{identidade});$$

$$\theta \cdot (\sigma \cdot q) = (\theta \circ \sigma) \cdot q \ (\textit{associatividade}).$$

O par $(Q, \cdot)$ é designado por *conjunto*-Π. □

Na sequência abreviaremos frequentemente $\sigma.q$ para q^σ e referir-nos-emos ao conjunto-Π $(Q, \cdot)$ apenas por Q, deixando a acção implícita. Se P for outro conjunto-Π, diz-se que P é um subconjunto-Π de Q se $P \subseteq Q$ e a acção em P for a restrição da acção em Q.

Um exemplo simples de um conjunto-Π é o próprio $\mathcal{N}$ com a acção $\sigma.x = \sigma(x)$. Qualquer conjunto A é um conjunto-Π com a acção $\sigma.a = a$ para todo o $\sigma \in \Pi$ e $a \in A$. Os exemplos mais importantes são as linguagens cujas expressões sintácticas são construídas com os nomes de $\mathcal{N}$. Nestes casos a acção de permutação consiste na substituição dos nomes nas expressões. No contexto deste trabalho vai ter particular interesse o conjunto dos termos do cálculo-π, que constitui um conjunto-Π conforme mostraremos na secção 4.3

2.2.2 Operações sobre conjuntos-Π

Nesta subsecção introduzem-se algumas operações sobre conjuntos-Π. Nomeadamente, como se estende a acção ao conjunto dos subconjuntos de um

conjunto-Π, ao produto cartesiano de conjuntos-Π, à soma de conjuntos-Π e ao espaço de funções definidas de um conjunto-Π em outro.

Potência de um conjunto-Π

Se Q é um conjunto-Π, define-se uma acção no conjunto potência de Q, $\mathcal{P}(Q)$, por $X^\sigma = \{x^\sigma : x \in X\}$ para $X \subseteq Q$. O conjunto das potências finitas de Q, $\mathcal{P}_{fin}(Q)$, é um subconjunto-Π de $\mathcal{P}(Q)$.

Produto e soma

Se P e Q são conjuntos-Π, O produto cartesiano $P \times Q$ é um conjunto-Π para a acção definida por $(p, q)^\sigma = (p^\sigma, q^\sigma)$ para $p \in P$ e $q \in Q$.

A soma $P + Q = (\{1\} \times P) \cup (\{2\} \times Q)$ é um conjunto-Π para a acção $(1, p)^\sigma = (1, p^\sigma)$, $(2, q)^\sigma = (2, q^\sigma)$ com $p \in P$ e $q \in Q$.

De futuro, para simplificar a notação, sempre que não houver perigo de confusão vamos escrever simplesmente p e q para nos referirmos aos elementos $(1, p)$ e $(2, q)$, respectivamente.

Funções

O conjunto $[P \to Q]$ de todas as funções de P em Q é um conjunto-Π onde, para $f : P \to Q$ e $\sigma \in \Pi$, tem-se f^σ definida por $f^\sigma(p) = f(p^{\sigma^{-1}})^\sigma$ para todo o $p \in P$. Se, para todo o $\theta \in \Pi$, denotarmos por θ_P e θ_Q as funções (bijectivas) $p \mapsto p^\theta$ e $q \mapsto q^\theta$ em P e em Q respectivamente, temos $f^\sigma = \sigma_Q \circ f \circ \sigma_P^{-1}$. Em termos de grafos, se o grafo de f é formado por todos os pares (p, q) tais que $p \in P$ e $q = f(p)$, o grafo de f^σ é formado por todos os pares (p^σ, q^σ) tais que (p, q) está no grafo de f.

2.2.3 Conjunto suporte

Aqui define-se a noção de suporte para os elementos de um conjunto-Π e apresentam-se alguns resultados válidos para conjuntos-Π em que todos os

elementos têm suporte finito - conjuntos nominais.

Definição 2.2.3 (Suporte, conjunto nominal) Seja Q um conjunto-Π.

1. Um conjunto $X \subseteq \mathcal{N}$ *suporta* $q \in Q$ se $q^\sigma = q$ para todo o $\sigma \in \Pi_X$.
2. Um elemento $q \in Q$ é *finitamente suportado* se q é suportado por um conjunto finito.
3. Dizemos que Q é um *conjunto nominal* se todos os seus elementos têm suporte finito.

□

De salientar que, como toda a permutação σ em Π_X é uma composição de transposições (xz) onde $x, z \notin X$, estas transposições também estão em Π_X. Assim, para verificar que X suporta q, temos apenas de verificar que $q^{(xz)} = q$ para todo o $x, z \notin X$.

Por definição de suporte, se X suporta q, então sempre que uma permutação σ é a identidade em X tem-se $q^\sigma = q$. Um resultado mais geral é que $q^\sigma = q^\theta$ se σ e θ coincidem quando restringidas ao conjunto X. Para mostrar esta afirmação vamos verificar que $(q^\sigma)^{\theta^{-1}} = q$, para isso temos apenas de mostrar que $\theta^{-1} \circ \sigma$ é a identidade em X, isto é, $(x^\sigma)^{\theta^{-1}} = x$ para todo o $x \in X$. Mas isso é imediato uma vez que, por hipótese, $x^\sigma = x^\theta$ para todo o $x \in X$.

Proposição 2.2.4 *Seja Q um conjunto-Π. Todo o $q \in Q$ finitamente suportado tem um menor conjunto finito que o suporta, denotado por $sup(q)$. Ou seja, $sup(q)$ suporta q e, se X é um conjunto finito que suporta q, então $sup(q) \subseteq X$.*

Demonstração Para garantir o resultado basta demonstrarmos que se X e Y são dois conjuntos finitos que suportam q, a sua intersecção $X \cap Y$ ainda

suporta q. Para o efeito vamos mostrar que se $x, y \notin X \cap Y$, então $q^{(xy)} = q$. Caso $x, y \notin X$ ou $x, y \notin Y$, a conclusão é imediata porque X e Y suportam q. Caso contrário, um de entre x, y está em X e o outro está em Y (mas nenhum dos dois está em ambos os conjuntos X e Y, por hipótese), vamos assumir que $x \in X$ e $y \in Y$. Como X e Y são finitos, existe z que não está em $X \cup Y$. Tem-se, $(xy) = (xz) \circ (yz) \circ (xz)$ e, como (xz) e (yz) aplicam q em q, porque são a identidade em Y e X respectivamente, vem que (xy) aplica q em q, como pretendíamos. □

A caracterização de $sup(q)$ também pode ser feita directamente, como mostra o seguinte resultado demonstrado por Gabbay e Pitts em [GP02].

Proposição 2.2.5 ***(Gabbay & Pitts)*** *Seja Q um conjunto-Π. Todo o $q \in Q$ finitamente suportado tem um menor conjunto que o suporta, $sup(q)$, dado por*

$$sup(q) = \{x \in \mathcal{N} : q^{(xz)} \neq q \text{ para infinitos } z \in \mathcal{N}\}.$$

□

Como iremos ver mais adiante, $sup(q)$ formaliza a noção de nomes livres de q.

Proposição 2.2.6 *Seja Q um conjunto-Π, $q \in Q$, $X \subseteq Q$ e $\sigma, \theta \in \Pi$. Verificam-se as seguintes propriedades:*

1. *Se X suporta q, então X^σ suporta q^σ.*

   Deste modo, se q é finitamente suportado, q^σ é finitamente suportado.

   Em particular, o conjunto de todos os elementos de Q finitamente suportados é um subconjunto-Π de Q, logo é um conjunto nominal.

2. *Se q é finitamente suportado, $sup(q^\sigma) = sup(q)^\sigma$.*

Demonstração 1. Se θ é a identidade em X^σ, temos de mostrar que $(q^\sigma)^\theta = q^\sigma$, que é equivalente a $((q^\sigma)^\theta)^{\sigma^{-1}} = q$. Como X suporta q, basta mostrarmos que $\sigma^{-1} \circ \theta \circ \sigma$ é a identidade em X, isto é, $x^{\sigma^{-1} \circ \theta \circ \sigma} = x$ para todo o $x \in X$. Mas a afirmação é equivalente a $(x^\sigma)^\theta = x^\sigma$ para todo o $x \in X$, que é verdade porque, por hipótese, θ é a identidade em X^σ.

2. Pela primeira propriedade, $sup(q^\sigma) \subseteq sup(q)^\sigma$. Pela mesma propriedade, agora aplicada a q^σ e σ^{-1}, temos $sup(q) = sup((q^\sigma)^{\sigma^{-1}}) \subseteq sup(q^\sigma)^{\sigma^{-1}}$, e portanto $sup(q)^\sigma \subseteq sup(q^\sigma)$. □

Vejamos alguns exemplos. O suporte de cada nome x pertencente a $\mathcal{N}$ é $\{x\}$, logo $\mathcal{N}$ é um conjunto nominal. Todo o conjunto A onde σ actua como a identidade, é um conjunto nominal em que o suporte de cada elemento de A é o vazio. Consideremos agora um caso mais complexo: os termos do cálculo-λ. Se considerarmos a igualdade textual dos termos-λ, o suporte de cada termo é o conjunto dos nomes que nele ocorrem. Se, em vez disso, considerarmos que dois termos são iguais a menos de conversão-α, então o suporte de cada termo é o conjunto dos nomes que nele ocorrem livres.

2.2.4 Operações sobre conjuntos nominais

Aqui estendem-se as operações apresentadas na subsecção 2.2.2 aos conjuntos nominais, introduzindo as definições e adaptações necessárias.

Potência de um conjunto nominal

Dado um conjunto nominal Q e um conjunto $P \subseteq Q$, P poderá ser ou não finitamente suportado. No entanto, $\mathcal{P}_{fin}(Q)$ é sempre um conjunto nominal, como mostra a proposição que se segue.

Proposição 2.2.7 *Se P é um subconjunto finito de Q, então P é finitamente suportado e $sup(P)$ é a união de todos os $sup(p)$ para p pertencente a P.*

Demonstração Concluí-se facilmente que a união dos $sup(p)$ é finita e suporta P. Vamos verificar que é o menor conjunto que suporta P. Suponhamos que x está na união – portanto x está em $sup(p)$ para algum p – mas não está em $sup(P)$. Pela proposição 2.2.5 (Gabbay & Pitts), existem infinitos z tais que $p^{(xz)} \neq p$, por isso podemos escolher um que não pertença ao suporte de nenhum dos elementos de P. Deste modo, (xz) é a identidade em $sup(P)$, logo $P^{(xz)}$ deverá ser igual a P. Mas, pela proposição 2.2.6 $sup(p^{(xz)}) = sup(p)^{(xz)}$, portanto z está no suporte de $p^{(xz)}$ porque x está no suporte de p, logo $p^{(xz)}$ está $P^{(xz)}$ mas não está em P, o que contraria a hipótese de $P^{(xz)} = P$ e portanto x está também em $sup(P)$. □

Se P for um subconjunto infinito de Q, poderá não ter suporte finito e portanto $\mathcal{P}(Q)$ não é, em geral, um conjunto nominal. Por exemplo $\mathcal{N}$ tem suporte finito mas dado $X \subseteq \mathcal{P}(\mathcal{N})$, $sup(X) = X$ logo $\mathcal{P}(\mathcal{N})$ não é um conjunto nominal. Mas poderão existir subconjuntos infinitos que têm suporte finito. Por exemplo, seja $Q = \mathcal{P}_{fin}(\mathcal{N})$. O suporte de $X \subseteq_{fin} \mathcal{N}$ é X. Vamos fixar $x \in \mathcal{N}$ e seja P o conjunto de todos os $\{x, z\}$ para $z \in \mathcal{N}$. Neste caso $sup(P) = \{x\}$ e logo P é finitamente suportado.

Produto e soma

Se P e Q são conjuntos nominais, é imediato que $P \times Q$ e $P + Q$ são também conjuntos nominais, e os suportes de (p, q), $(1, p)$ e $(2, q)$ são respectivamente $sup(p) \cup sup(q)$, $sup(p)$ e $sup(q)$.

Funções

O conjunto-Π $[P \to Q]$ não é, em geral, um conjunto nominal. Mas, o conjunto de todas as funções, de P em Q, finitamente suportadas, que iremos denotar por Q^P, é um subconjunto-Π de $[P \to Q]$ e é um conjunto nominal. Temos a seguinte caracterização para o suporte de $[P \to Q]$.

Proposição 2.2.8 *O conjunto $X \subseteq \mathcal{N}$ suporta $f : P \to Q$ se, e só se,*

$$f(p^\sigma) = f(p)^\sigma$$

para todo o $p \in P$ e toda a permutação $\sigma \in \Pi_X$. Neste caso, $X \cup sup(p)$ suporta $f(p)$ para todo o $p \in P$.

Demonstração Afirmar que $f^\sigma = f$ é equivalente a afirmar que $f^{\sigma^{-1}} = f$, que por sua vez é o mesmo que $f(p^\sigma)^{\sigma^{-1}} = f(p)$ para todo o $p \in P$, isto é, $f(p^\sigma) = f(p)^\sigma$ para todo o $p \in P$. Como $X \subseteq \mathcal{N}$ suporta $f : P \to Q$ se, e só se, $f^\sigma = f$ para todo o $\sigma \in \Pi_X$, tem-se a conclusão pretendida. Quanto à segunda parte da proposição, atendendo a que se σ é a identidade em $X \cup sup(p)$, então para todo o $p \in P$ tem-se $f(p)^\sigma = f(p^\sigma) = f(p)$, o que mostra que $X \cup sup(p)$ suporta $f(p)$. □

No decorrer do trabalho estaremos interessados apenas no caso em que $P = \mathcal{N}$. Uma função $f : \mathcal{N} \to Q$ será interpretada como ou um sistema que recebe um nome x e continua como $f(x)$, ou então um sistema que gera um nome novo (que não ocorre livre no sistema inicial), x, e continua como $f(x)$. No primeiro caso x é arbitrário porque é fornecido pelo exterior; no segundo caso, o facto de ser novo significa que x não está no suporte de f. Vamos designar os elementos de $Q^{\mathcal{N}}$ por *funções de abstracção*.

Teorema 2.2.9 *Um conjunto $X \subseteq \mathcal{N}$ suporta $f : \mathcal{N} \to Q$ se, e só se, $X \cup \{x\}$ suporta $f(x)$ para todo o $x \in \mathcal{N}$, e $f(z) = f(x)^{(xz)}$ (relação de generalidade) para todo o $x, z \notin X$.*

Demonstração Suponhamos, em primeiro lugar, que X suporta f. Se σ é a identidade em $X \cup \{x\}$ para todo o $x \in \mathcal{N}$, então $f^\sigma = f$ e $x^{\sigma^{-1}} = x$ logo $f(x)^\sigma = f(x^{\sigma^{-1}})^\sigma = f^\sigma(x) = f(x)$, o que mostra que $X \cup \{x\}$ suporta $f(x)$. Se $x, z \notin X$, e atendendo a que $(xz)^{-1} = (xz)$, temos $f(x)^{(xz)} = f(z^{(xz)})^{(xz)} = f^{(xz)}(z) = f(z)$, porque (xz) é a identidade em

X. Reciprocamente, temos de mostrar que se σ é a identidade em X, então $f^\sigma = f$. Como toda a permutação é uma composição de transposições, é suficiente mostrar que $f^{(xz)} = f$ para todo o $x, z \notin X$. Temos $f^{(xz)}(z) = f(z^{(xz)}) = f(x)^{(xz)} = f(z)$ por hipótese, e de modo análogo se conclui que $f^{(xz)}(x) = f(x)$. Para $y \neq x, z$, a permutação (xz) é a identidade em $X \cup \{y\}$, portanto $f^{(xz)}(y) = f(y^{(xz)}) = f(y)^{(xz)} = f(y)$, visto que $X \cup \{y\}$ suporta $f(y)$. □

2.2.5 Morfismos de conjuntos-Π

Aqui definem-se os morfismos para os conjuntos-Π e em particular para os conjuntos nominais, e conclui-se que os conjuntos nominais constituem uma categoria.

Definição 2.2.10 (Morfismo) Um *morfismo* de conjuntos-Π $f : P \to Q$, é uma função tal que $f(\sigma \cdot p) = \sigma \cdot f(p)$ para todo o $\sigma \in \Pi$ e $p \in P$. Usando a notação exponencial, a condição toma a forma $f(p^\sigma) = f(p)^\sigma$. Um morfismo de conjuntos nominais é um morfismo dos correspondentes conjuntos-Π. □

Aos morfismos de conjuntos-Π também é costume dar o nome de funções *equivariantes*. Note-se que, como elemento de $[P \to Q]$, um morfismo $f : P \to Q$ satisfaz $sup(f) = \emptyset$, como é fácil verificar.

O resultado seguinte mostra que um morfismo de conjuntos-Π se restringe a um morfismo dos subconjuntos-Π dos elementos de suporte finito.

Proposição 2.2.11 *Sejam P e Q conjuntos-Π e $f : P \to Q$ um morfismo. Se $p \in P$ é finitamente suportado, $f(p)$ é finitamente suportado e $sup(f(p)) \subseteq sup(p)$.*

Demonstração Se σ é a identidade em $sup(p)$, então $f(p)^\sigma = f(p^\sigma) = f(p)$. Logo, $sup(p)$ suporta $f(p)$ e portanto $sup(f(p)) \subseteq sup(p)$. □

O resultado que se segue foi apresentado por Kohei Honda em [Hon00].

Proposição 2.2.12 ***(Honda)*** *Sejam P e Q conjuntos nominais e f um morfismo de P em Q. Se $q \in Q$ está na imagem de f, então*

$$sup(q) = \bigcap \{sup(p) : f(p) = q\}.$$

Em particular, se f é injectiva, $sup(f(p)) = sup(p)$ para todo o $p \in P$.

Demonstração Se $f(p) = q$, então, pela proposição anterior, $sup(q) \subseteq sup(p)$. Reciprocamente, suponhamos $x \notin sup(q)$. Temos de mostrar que $x \notin sup(p)$ para algum p tal que $f(p) = q$. É suficiente mostrarmos que se x está no suporte de algum p_0, é possível encontrar outro p cujo suporte não contém x. Escolhemos $a \notin sup(p_0) \cup sup(q)$ e fazemos $p = p_0^{(ax)}$. Nestas condições $f(p) = f(p_0^{(ax)}) = f(p_0)^{(ax)} = q^{(ax)} = q$ e $x \notin sup(p_0)^{(ax)} = sup(p_0^{(ax)}) = sup(p)$, como pretendíamos. □

Note-se que a igualdade $sup(f(p)) = sup(p)$ nem sempre se verifica se f não é injectiva.

Naturalmente que a identidade é um morfismo pois $id_{\mathcal{N}}(n^\sigma) = n^\sigma = id_{\mathcal{N}}(n)^\sigma$. Dados f e g morfismos de conjuntos-Π, verifica-se facilmente que a composição de morfismos é ainda um morfismo: $(g \circ f)(p^\sigma) = g(f(p^\sigma))) = g(f(p)^\sigma) = g(f(p))^\sigma = (g \circ f)(p)^\sigma$. Assim, os conjuntos nominais e os seu morfismos constituem uma categoria.

2.2.6 Functores

Nesta secção vamos apresentar vários functores que integram a construção do functor principal que será, mais tarde, utilizado na definição do domínio semântico do cálculo-π. Os functores apresentados nesta secção supõem-se todos definidos na categoria dos conjuntos nominais e seus morfismos.

O functor constante $\mathcal{N}$

Este functor aplica cada conjunto nominal Q em $\mathcal{N}$ e aplica cada morfismo $f : P \to Q$ na identidade em $\mathcal{N}$. O functor será também denotado por $\mathcal{N}$.

Produto

A construção usual de produto de conjuntos é um functor entre conjuntos nominais. Se $f : P \to P'$ e $g : Q \to Q'$ são morfismos, a função $f \times g : P \times Q \to P' \times Q'$ é dada por $(f \times g)(p, q) = (f(p), g(q))$. Esta função é ainda um morfismo pois

$$\begin{aligned}(f \times g)((p, q)^\sigma) &= (f \times g)(p^\sigma, q^\sigma) \\ &= (f(p^\sigma), g(q^\sigma)) \\ &= (f(p)^\sigma, g(q)^\sigma) \\ &= (f(p), g(q))^\sigma \\ &= (f \times g)(p, q)^\sigma.\end{aligned}$$

Soma

Tal como no caso do produto, a soma usual de conjuntos também constitui um functor entre conjuntos nominais. Assim, se $f : P \to P'$ e $g : Q \to Q'$ são morfismos, a função $f + g : P + Q \to P' + Q'$ aplica cada $(1, p)$ em $(1, f(p))$ e cada $(2, q)$ em $(2, g(q))$. Verifica-se facilmente que $f + g$ é um morfismo de conjuntos nominais:

$$\begin{aligned}(f + g)(i, s^\sigma) &= \begin{cases} (1, f(s^\sigma)), & \text{se } i = 1 \\ (2, g(s^\sigma)), & \text{se } i = 2 \end{cases} \\ &= \begin{cases} (1, f(s)^\sigma), & \text{se } i = 1 \\ (2, g(s)^\sigma), & \text{se } i = 2 \end{cases} \\ &= (f + g)(i, s)^\sigma.\end{aligned}$$

Funções finitamente suportadas

Vamos considerar aqui o functor que aplica cada conjunto nominal Q no conjunto $Q^{\mathcal{N}}$ das funções finitamente suportadas de $\mathcal{N}$ em Q, e cada morfismo $f : P \to Q$ no morfismo $f^{\mathcal{N}} : P^{\mathcal{N}} \to Q^{\mathcal{N}}$ dado por $f^{\mathcal{N}}(t) = f \circ t$ para toda a $t : \mathcal{N} \to P$ finitamente suportada.

Temos de verificar que $f^{\mathcal{N}}$ é de facto um morfismo, isto é que $f^{\mathcal{N}}(t^\sigma) = f^{\mathcal{N}}(t)^\sigma$ o que é o mesmo que $f \circ t^\sigma = (f \circ t)^\sigma$, para todo o t e σ. Ora, para $x \in \mathcal{N}$, tem-se $f(t^\sigma(x)) = f(t(x^{\sigma^{-1}})^\sigma) = f(t(x^{\sigma^{-1}}))^\sigma = (f \circ t)(x^{\sigma^{-1}})^\sigma = (f \circ t)^\sigma(x)$.

Teríamos ainda de verificar que $f \circ t$ é finitamente suportada, mas isso decorre do facto de $f^{\mathcal{N}}$ ser um morfismo.

Potência de suporte finito

O functor potência de suporte finito, $\mathcal{P}_{sf}$, associa a cada conjunto nominal Q o conjunto nominal $\mathcal{P}_{sf}(Q)$ dos subconjuntos de Q finitamente suportados, e a cada morfismo $f : P \to Q$, o morfismo $\mathcal{P}_{sf}(f) : \mathcal{P}_{sf}(P) \to \mathcal{P}_{sf}(Q)$ que a cada X, subconjunto finitamente suportado de P, faz corresponder o conjunto $\{f(x) : x \in X\}$. Vamos verificar que $\mathcal{P}_{sf}(f)$ é de facto um morfismo: $\mathcal{P}_{sf}(f)(X^\sigma) = \{f(x^\sigma) : x \in X\} = \{f(x)^\sigma : x \in X\} = \{f(x) : x \in X\}^\sigma = \mathcal{P}_{sf}(f)(X)^\sigma$. Isto também implica que $\mathcal{P}_{sf}(f)(X)$ é finitamente suportado, como pretendíamos.

O functor $\mathcal{P}_{sf}$ vai ter apenas um papel auxiliar na construção do domínio semântico do cálculo-π, o nosso interesse principal irá centrar-se no subfunctor das potências finitas de um conjunto, $\mathcal{P}_{fin}$.

Um functor para a semântica do Cálculo-π

Adiante, para definir a semântica do Cálculo-π, iremos considerar, na categoria dos conjuntos nominais, o endofunctor definido por:

$$F(Q) = \mathcal{P}_{fin}(Q + (\mathcal{N} \times \mathcal{N} \times Q) + (\mathcal{N} \times Q^{\mathcal{N}}) + (\mathcal{N} \times Q^{\mathcal{N}}))$$

onde $Q^{\mathcal{N}}$ é o conjunto das funções de $\mathcal{N}$ em Q de suporte finito. Ora, utilizando os resultados apresentados para os functores que compõem F, se $f : Q \to P$ é um morfismo de conjuntos nominais, a função $F(f) : F(Q) \to F(P)$ é dada por

$$\begin{aligned} F(f)(X) = \; & \{f(q) : q \in X\} \\ & \cup \\ & \{(\overline{a}, x, f(q)) : (\overline{a}, x, q) \in X\} \\ & \cup \\ & \{(\overline{a}, f \circ e) : (\overline{a}, e) \in X\} \\ & \cup \\ & \{(a, f \circ t) : (a, t) \in X\} \end{aligned}$$

O functor F é monótono pois se tomarmos $S \subseteq Q$ temos também $(\mathcal{N} \times \mathcal{N} \times S) \subseteq (\mathcal{N} \times \mathcal{N} \times Q)$ e se atendermos a que toda a função de $S^{\mathcal{N}}$ se pode estender de uma forma única a uma função $Q^{\mathcal{N}}$ (basta alterar-lhe o contradomínio) e denotarmos por $[S^{\mathcal{N}}]_Q$ esta extensão, temos $[S^{\mathcal{N}}]_Q \subseteq Q^{\mathcal{N}}$. Identificando cada uma das funções de $S^{\mathcal{N}}$ com a correspondente função em $[S^{\mathcal{N}}]_Q$ temos $\mathcal{N} \times S^{\mathcal{N}} \subseteq \mathcal{N} \times Q^{\mathcal{N}}$. Consequentemente $F(S) \subseteq F(Q)$ uma vez que o functor $\mathcal{P}_{fin}$ preserva a inclusão.

2.3 Coálgebras

As coálgebras são muito utilizadas nas ciências da computação para descrever sistemas dinâmicos em que o espaço dos estados não é observável directamente, podemos apenas observar as operações efectuadas sobre este mesmo espaço, [RT94, Rut96, JR97]. Neste contexto as técnicas de coindução são fundamentais tanto para as definições como para as demonstrações. Igualmente importantes são as bissimulações, isto é, elementos que do ponto de vista observacional são indistinguíveis.

A utilização de coálgebras nas ciências da computação começou com Aczel e a teoria dos conjuntos não bem-fundados e estendeu-se depois à

teoria dos autómatos e à semântica, especificação e verificação de programas concorrentes e orientados para objectos. As coálgebras são uma estrutura matemática simples mas fundamental para captar o comportamento de sistemas dinâmicos. Muitas das noções teóricas fundamentais dos sistemas, como a invariância e bissimulação, podem ser descritas com base na teoria das coálgebras. O elemento fundamental é a utilização das coálgebras finais para captar o comportamento (possivelmente infinito) dos sistemas. Estas coálgebras finais podem ser obtidas como uma generalização dos maiores pontos fixos em contraste com os menores pontos fixos utilizados para obter álgebras iniciais. Uma poderosa técnica de prova nesse contexto de finalidade é a co-indução que se baseia na noção de bissimulação, introduzida no contexto da semântica da concorrência por Milner, [Mil80], para formalizar a equivalência de comportamento entre processos concorrentes. A bissimulação foi mais tarde introduzida na teoria das coálgebras por Aczel e Mendler, [AM88], que criaram uma definição categorial que se aplica a coálgebras arbitrárias. Usando esta noção de bissimulação Aczel [Acz88] formulou o principio da coindução de forma muito semelhante ao modo como Milner introduziu o seu "método e prova por bissimulação": para provar que dois processos tem comportamento equivalente (bissimilar), basta provar a existência de uma relação de bissimulação entre eles. Este princípio de coindução assume particular interesse no contexto das coálgebras finais, porque numa coálgebra final a bissimilaridade coincide com a identidade e portanto prova-se a igualdade construindo relações de bissimulação. O estudo das coálgebras nas linhas da Álgebra Universal foi iniciado por Rutten em [Rut95] e [Rut96] (agora com uma versão mais recente [Rut00]). Encontram-se cada vez mais aplicações para as coálgebras em vários ramos da matemática e das ciências da computação e esta teoria está a tornar-se bastante promissora.

2.3.1 Conceitos básicos

Vamos começar por introduzir algumas definições básicas e em seguida os produtos fibrados, as bissimulações e as bissimilaridades.

Definição 2.3.1 (Coálgebra) Dado um functor F, uma *coálgebra-F* ou simplesmente *coálgebra* é um par (S, α) onde S é um conjunto e α uma função de S em $F(S)$. □

O conjunto S é usualmente designado por *suporte* e a função α por *estrutura* ou *operação* da coálgebra. Quando as coálgebras são utilizadas para descrever algum tipo de sistema dinâmico o conjunto suporte, S, também é designado por *espaço de estados*.

Exemplos 2.3.2

1. Consideremos o caso de um autómato finito com saída (modelo de Moore):

Dados um conjunto de entrada A e um conjunto de saída B, um autómato sobre A e B é um triplo (Q, δ, β), onde Q é um conjunto de estados, $\delta : Q \times A \to Q$ é uma função de transição e $\beta : Q \to B$ é uma função de saída. Podemos reunir as duas funções numa só. Primeiro transforma-se δ em $\hat{\delta} : Q \to Q^A$, em que Q^A é o conjunto de todas as funções de A em Q, fazendo $\hat{\delta}(q)(a) = \delta(q, a)$ para todos os $q \in Q$, $a \in A$. As funções $\hat{\delta}$ e β têm agora o mesmo domínio e podem reunir-se numa só função $(\hat{\delta}, \beta) : Q \to Q^A \times B$ definida como habitualmente por $(\hat{\delta}, \beta)(q) = (\hat{\delta}(q), \beta(q))$ para todo o $q \in Q$. Podemos agora definir o autómato como uma coálgebra (Q, ϕ) para o functor $F = (-)^A \times B$ em que, para todo $q \in Q$, $\phi(q) = (\hat{\delta}(q), \beta(q))$. A interpretação para $\phi(q) = (p, b)$ é que no estado q o autómato passa para o estado p, que é função do valor de entrada, produzindo como saída b.

2. Vejamos agora o caso de um sistema de transições não determinista:

Dado um conjunto de acções A, um sistema de transições sobre A é um par $(S, \to)$ onde S é um conjunto de estados e $\to \subseteq S \times A \times A$ é uma relação. É usual abreviar $(s, a, t) \in \to$ por $s \xrightarrow{a} t$ que é interpretado como "no estado

s o sistema de transições observa (ou executa) a acção a e passa para o estado t". Definindo $\phi : S \to \mathcal{P}(A \times S)$, onde $\mathcal{P}$ é o functor potência, por $\phi(s) = \{(a,t) : s \xrightarrow{a} t\}$, tem-se (S,ϕ) uma coálgebra para o functor $F = \mathcal{P}(A \times (-))$. A interpretação para $(a,t) \in \phi(s)$ é: no estado s pode observar-se a acção a e passar para o estado t. A interpretação para $\phi(s) = \emptyset$ é: não existem transições a partir do estado s. □

A definição de coálgebra pode ser feita para um endofunctor em qualquer categoria e não apenas na categoria dos conjuntos. Para além de uma breve referência à categoria CCfe como categoria de base, na aplicação à semântica do cálculo-π estaremos interessados em endofunctores na categoria dos conjuntos nominais, definida mais adiante.

Definição 2.3.3 (Homomorfismo de coálgebras) Seja F um functor. Um *homomorfismo de coálgebras* de uma coálgebra-F, (S,α), em outra coálgebra-F, (T,β), é uma função entre os conjuntos suporte, $f : S \to T$, tal que $\beta \circ f = F(f) \circ \alpha$, isto é, tal que o diagrama que se segue comuta.

$$\begin{array}{ccc} S & \xrightarrow{f} & T \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ F(S) & \xrightarrow[F(f)]{} & F(T) \end{array}$$

□

Dados dois homomorfismos, f e g, entre coálgebras-F, a sua composição $g \circ f$ (caso seja possível) é ainda um homomorfismo de coálgebras-F.

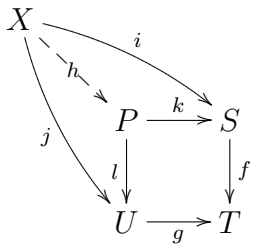
Definição 2.3.4 (Coálgebra final) Seja F um functor. Uma *coálgebra final*, $\gamma : T \to F(T)$, é uma coálgebra tal que para toda a coálgebra do mesmo tipo, $\alpha : S \to F(S)$, existe um único morfismo de coálgebras de (S,α) em (T,γ). □

Uma coálgebra-F final (T, γ) é um ponto fixo de F, isto é, $\gamma : T \to F(T)$ é um isomorfismo, [RT94]. As coálgebras finais quando existem, são únicas a menos de isomorfismo. Em [Rut00] existem teoremas mais gerais que garantem a existência de coálgebras finais para certas classes de functores (polinomiais, limitadas). Não iremos, contudo, utilizar esses resultados porque optamos por construir explicitamente as coálgebras finais de que iremos necessitar.

De notar que, na sequência do que vimos, na secção 2.1 para as equações de domínios sobre cfe's, temos o seguinte corolário do teorema 5.31, apresentado em [Mon98], que estabelece que todo o ponto fixo de um functor localmente aproximante é uma coálgebra final.

Corolário 2.3.5 *Seja F um functor na categoria dos* CCfe. *Se $\gamma : T \to F(T)$ é um isomorfismo então (T, γ) é uma coálgebra final.* □

Definição 2.3.6 (Produto fibrado) O *produto fibrado* das funções $f : S \to T$ e $g : U \to T$ é um triplo (P, k, l),onde $k : P \to S$ e $l : P \to U$ são morfismos que satisfazem $f \circ k = g \circ l$, tal que para todo o conjunto X e morfismos $i : X \to S$ e $j : X \to U$ com $f \circ i = g \circ j$ existe um único morfismo $h : X \to P$ com $k \circ h = i$ e $l \circ h = j$.



Os morfismos k e l são designados por morfismos projecção. O *produto fibrado fraco* define-se do mesmo modo mas sem exigir que a função h seja única. □

As relações compatíveis com a estrutura de coálgebra são designadas por bissimulações. A designação está relacionada com o caso particular dos sis-

temas de transições: se um estado s_1 de um sistema S_1 simula um estado s_2 de um sistema S_2 e vice-versa dizemos que s_1 e s_2 são bissimilares. Uma bissimulação é definida como um subconjunto particular desses pares bissimilares.

Definição 2.3.7 (Bissimulação) Sejam (S, α) e (T, β) coálgebras-F. A relação $R \subseteq S \times T$ é uma *bissimulação* entre S e T se existir $\rho : R \to F(R)$ (não necessariamente única) tal que as projecções $\pi_1 : R \to S$ e $\pi_2 : R \to T$ são homomorfismos relativamente à coálgebra (R, ρ). Uma bissimulação em S é uma bissimulação de S em S. □

Assim, em Set uma bissimulação entre S e T é uma relação binária $R \subseteq S \times T$ para a qual existe ρ, coálgebra-F, tal que o seguinte diagrama comuta

$$\begin{array}{ccccc} S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & T \\ {\scriptstyle\alpha}\downarrow & & {\scriptstyle\rho}\downarrow & & \downarrow{\scriptstyle\beta} \\ F(S) & \xleftarrow[F(\pi_1)]{} & F(R) & \xrightarrow[F(\pi_2)]{} & F(T) \end{array}$$

Definição 2.3.8 (Bissimilaridade) Sejam s e t elementos de coálgebras-F. Diz-se que s e t são *bissimilares*, e representa-se por $s \dot{\sim} t$ se existe uma relação de bissimulação, R, tal que $(s, t) \in R$. □

Portanto, a bissimilaridade ($\dot{\sim}$) é a união de todas as bissimulações e é ainda uma bissimulação.

Numa coálgebra final a relação de bissimilaridade coincide com a relação de igualdade, [RT94]. Assim para provar a igualdade de dois elementos basta verificar que estes são bissimilares - este princípio é usualmente conhecido como o *princípio da co-indução*.

2.3.2 Caracterização da bissimilaridade

A proposição 2.3.9 será usada no capítulo 4 para provar que a equivalência semântica forte do cálculo-π coincide com a bissimilaridade forte. Esta secção demonstra esse resultado e mostra que o functor que irá ser usado no capítulo 4 satisfaz duas das condições da proposição. Alguns dos resultados aqui apresentados são baseados em resultados apresentados por Rutten em [Rut00], com as respectivas demonstrações adaptadas à categoria dos conjuntos nominais.

Nesta secção iremos assumir que todos os functores são endofunctores na categoria dos conjuntos-Π nominais.

Proposição 2.3.9 *Seja F um functor e f um morfismo de coálgebras-F. Suponhamos que se verificam as seguintes propriedades:*

1. *F possui uma coálgebra final $\mathbb{T}$.*
2. *O núcleo de equivalência de f, $E_f = \{(s,t) : f(s) = f(t)\}$, é uma bissimulação.*
3. *A imagem $\{(f(s), f(t)) : (s,t) \in R\}$ de uma bissimulação R por f é uma bissimulação.*

Então a relação de bissimilaridade de qualquer coálgebra-F coincide com o núcleo de equivalência do único morfismo da coálgebra dada na coálgebra final.

Demonstração Seja f o único morfismo da coálgebra dada em $\mathbb{T}$. A segunda condição da proposição implica que E_f está contido na relação de bissimilaridade $\dot{\sim}$. Mas a imagem de $\dot{\sim}$ por f é uma bissimulação em $\mathbb{T}$. Como a bissimilaridade em $\mathbb{T}$ é a relação identidade, por $\mathbb{T}$ ser final , tem-se que $f(s) = f(t)$ sempre que $s \dot{\sim} t$. Mostra-se assim que $\dot{\sim}$ está contida em E_f. □

Vamos agora verificar que o functor que vamos utilizar para definir a semântica do cálculo-π, $F = \mathcal{P}_{fin}(Id+(\mathcal{N}\times\mathcal{N}\times Id)+(\mathcal{N}\times Id^{\mathcal{N}})+(\mathcal{N}\times Id^{\mathcal{N}}))$, introduzido na secção 2.2.6, satisfaz as duas últimas condições da proposição. Quanto à primeira condição, iremos mostrar, no capítulo 4, que F possui uma coálgebra final. Para a verificação da terceira condição, vamos utilizar o seguinte resultado apresentado (e demonstrado) por J. Rutten no seu artigo sobre coálgebras universais [Rut00].

Proposição 2.3.10 *Se $f : T \to S$ e $g : T \to U$ são morfismos de coálgebras, então $\langle f, g\rangle(T) = \{(f(t), g(t)) : t \in T\}$ é uma bissimulação de S e U.* □

Passemos então à demonstração da terceira condição para o functor em questão.

Proposição 2.3.11 *Seja $f : S \to T$ um morfismo de coálgebras e R uma bissimulação em S. A imagem $\{(f(s), f(t)) : (s, t) \in R\}$ de R por f é uma bissimulação.*

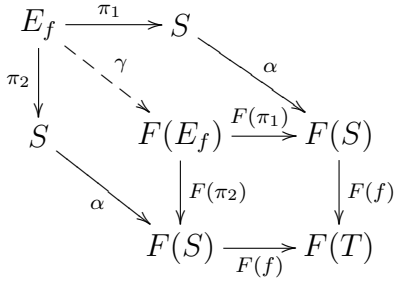
Demonstração Como R é uma bissimulação, existe $\gamma : R \to F(R)$ tal que as projecções $\pi_1, \pi_2 : R \to S$ são morfismos de coálgebras. Mas neste caso, $f \circ \pi_1, f \circ \pi_2 : R \to T$ são também morfismos de coálgebras (a composição de morfismos é ainda um morfismo), e a imagem de R é $(f \circ \pi_1, f \circ \pi_2)(R)$, que é uma bissimulação atendendo à proposição anterior. □

Falta-nos apenas demonstrar a segunda condição, para a qual é necessário que F preserve produtos fibrados fracos. Recordemos a definição: um functor F preserva produtos fibrados fracos se o seguinte diagrama da esquerda é um produto fibrado fraco sempre que o diagrama da direita o é:

$$\begin{array}{ccc} F(W) & \xrightarrow{F(p)} & F(X) \\ {\scriptstyle F(q)}\downarrow & & \downarrow{\scriptstyle F(f)} \\ F(Y) & \xrightarrow[F(g)]{} & F(Z) \end{array} \qquad \begin{array}{ccc} W & \xrightarrow{p} & X \\ {\scriptstyle q}\downarrow & & \downarrow{\scriptstyle f} \\ Y & \xrightarrow[g]{} & Z \end{array}$$

Proposição 2.3.12 *Se um functor F preserva produtos fibrados fracos, o núcleo de equivalência de todo o morfismo de coálgebras-F é uma bissimulação.*

Demonstração Sejam (S, α) e (T, β) coálgebras, $f : S \to T$ um morfismo e E_f o núcleo de equivalência de f. Se $f(s) = f(t)$, então $f(s^\sigma) = f(s)^\sigma = f(t)^\sigma = f(t^\sigma)$, logo $(s,t) \in E_f$ implica $(s,t)^\sigma \in E_f$, e portanto E_f é um conjunto nominal. Mais, $S \overset{\pi_2}{\leftarrow} E_f \overset{\pi_1}{\rightarrow} S$ é um produto fibrado de $S \overset{f}{\rightarrow} T \overset{f}{\leftarrow} S$, onde π_1 e π_2 são a primeira e a segunda projecção,respectivamente. Definimos a coálgebra (E_f, γ) pelo seguinte diagrama:



Para ver que esta definição faz sentido, comecemos por verificar que o hexágono exterior comuta:

$$\begin{aligned} F(f) \circ \alpha \circ \pi_1 &= \beta \circ f \circ \pi_1 \\ &= \beta \circ f \circ \pi_2 \\ &= F(f) \circ \alpha \circ \pi_2. \end{aligned}$$

A primeira e a última igualdade são justificadas pelo facto de f ser um morfismo ($F(f) \circ \alpha = \beta \circ f$) e igualdade intermédia deve-se ao facto de E_f, π_1, π_2 ser um produto fibrado ($f \circ \pi_1 = f \circ \pi_2$). A preservação dos produtos fibrados garante a existência de γ. A comutatividade dos quadrados superiores

garante que E_f é uma bissimulação. □

Proposição 2.3.13 *O functor*

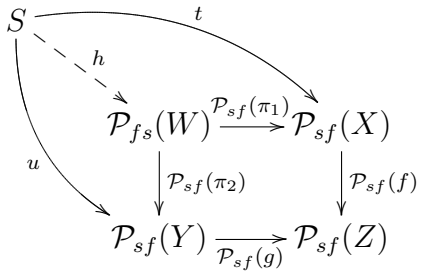
$$F = \mathcal{P}_{fin}(Id + (\mathcal{N} \times \mathcal{N} \times Id) + (\mathcal{N} \times Id^{\mathcal{N}}) + (\mathcal{N} \times Id^{\mathcal{N}}))$$

preserva produtos fibrados fracos.

Demonstração A forma mais simples de assegurar o resultado é mostrar que os functores que compõem F preservam produtos fibrados fracos. Vamos começar por mostrar em pormenor que $\mathcal{P}_{fin}$ ou, de modo mais geral $\mathcal{P}_{sf}$, e $Id^{\mathcal{N}}$ preservam produtos fibrados fracos. Para os restantes functores que compõem F, vamos apenas apresentar a definição da função h pois a partir daí o resultado mostra-se de modo semelhante.

$\mathcal{P}_{sf}$ preserva produtos fibrados fracos

Consideremos o diagrama:



Temos de mostrar que se $\mathcal{P}_{sf}(f) \circ t = \mathcal{P}_{sf}(g) \circ u$, existe uma função (não necessariamente única) $h : S \to \mathcal{P}_{sf}(W)$ tal que os triângulos comutam. Sabemos que $W = \{(x, y) \in X \times Y : f(x) = g(y)\}$. Defina-se h por

$$h(s) = \{w \in W : \pi_1(w) \in t(s),\, \pi_2(w) \in u(s)\}.$$

Temos de verificar os seguintes pontos:

- Para todo o s, $h(s)$ está em $\mathcal{P}_{sf}(W)$, isto é, $h(s)$ é finitamente suportada:

  Vamos mostrar que se $M \subseteq \mathcal{N}$ suporta $t(s)$ e $N \subseteq \mathcal{N}$ suporta $u(s)$, então $M \cup N$ suporta $h(s)$. Vamos assumir que σ é a identidade em $M \cup N$ e temos de mostrar que $h(s)^\sigma = h(s)$. Por definição de h, esta condição é equivalente a termos $w^\sigma \in h(s)$ para todo o $w \in h(s)$. Ora se $w \in h(s)$, então $\pi_1(w) \in t(s)$, logo $\pi_1(w^\sigma) = \pi_1(w)^\sigma \in t(s)^\sigma = t(s)$. De modo similar, como $\pi_2(w^\sigma) \in u(s)$, então $w^\sigma \in h(s)$, como pretendíamos.

- A função h é um morfismo, isto é, $h(s^\sigma) = h(s)^\sigma$ para toda a permutação σ:

$$\begin{aligned} h(s)^\sigma &= \{w^\sigma : w \in W, \pi_1(w) \in t(s), \pi_2(w) \in u(s)\} \\ &= \{w^\sigma : w \in W, \pi_1(w)^\sigma \in t(s)^\sigma, \pi_2(w)^\sigma \in u(s)^\sigma\} \\ &= \{w^\sigma : w \in W, \pi_1(w^\sigma) \in t(s^\sigma), \pi_2(w^\sigma) \in u(s^\sigma)\} \\ &= \{w \in W : \pi_1(w) \in t(s^\sigma), \pi_2(w) \in u(s^\sigma)\} \\ &= h(s^\sigma). \end{aligned}$$

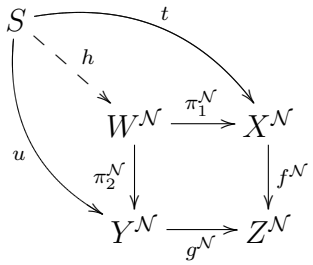
  A penúltima igualdade é devida ao facto que σ aplica bijectivamente W em si mesmo porque W é um conjunto nominal.

- Os triângulos comutam, isto é, $\mathcal{P}_{sf}(\pi_1)(h(s)) = t(s)$ e $\mathcal{P}_{sf}(\pi_2)(h(s)) = u(s)$:

  Temos $\mathcal{P}_{sf}(\pi_1)(h(s)) = \{\pi_1(w) : w \in h(s)\} = \{\pi_1(w) : w \in W, \pi_1(w) \in t(s), \pi_2(w) \in u(s)\} \subseteq t(s)$. Se $x \in t(s)$, então, como $\mathcal{P}_{sf}(f) \circ t = \mathcal{P}_{sf}(g) \circ u$, existe $y \in u(s)$ tal que $f(x) = g(y)$. Como W produto fibrado fraco, existe $w \in W$ tal que $\pi_1(w) = x$ e $\pi_2(w) = y$. Logo $x \in t(s)$. De modo idêntico prova-se que $\mathcal{P}_{sf}(\pi_2)(h(s)) = u(s)$.

$Id^{\mathcal{N}}$ preserva produtos fibrados fracos

Consideremos o diagrama:



Temos de mostrar que se $f^{\mathcal{N}} \circ t = g^{\mathcal{N}} \circ u$, existe uma função (não necessariamente única) $h : S \to W^{\mathcal{N}}$ tal que os triângulos superiores comutam. Sabemos que $W = \{(x, y) \in X \times Y : f(x) = g(y)\}$. Defina-se h por

$$h(s) = (t(s), u(s)).$$

Temos de verificar os seguintes pontos:

- Para todo o s, $h(s)$ está em $W^{\mathcal{N}}$, isto é, $h(s)$ é uma função de $\mathcal{N}$ em W com suporte finito. Ora como $t(s)$ e $u(s)$ têm suporte finito, vamos mostrar que se $M \subseteq \mathcal{N}$ suporta $t(s)$ e $N \subseteq \mathcal{N}$ suporta $u(s)$, então $M \cup N$ suporta $h(s)$. Vamos assumir que σ é a identidade em $M \cup N$ e temos de mostrar que $h(s)^\sigma = h(s)$. Ora, por definição de h, $h(s)^\sigma = (t(s), u(s))^\sigma = (t(s)^\sigma, u(s)^\sigma) = (t(s), u(s)) = h(s)$, como pretendíamos.

- A função h é um morfismo, isto é, $h(s^\sigma) = h(s)^\sigma$ para toda a permutação σ. Para todo o $n \in \mathcal{N}$ temos

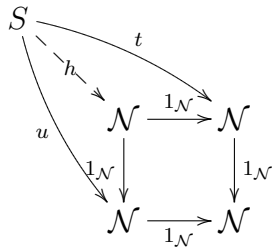
$$\begin{aligned} h(s)^\sigma(n) &= h(s)(n^{\sigma^{-1}})^\sigma \\ &= (t(s)(n^{\sigma^{-1}}), u(s)(n^{\sigma^{-1}}))^\sigma \\ &= (t(s)(n^{\sigma^{-1}})^\sigma, u(s)(n^{\sigma^{-1}})^\sigma) \\ &= (t(s)^\sigma(n), u(s)^\sigma(n)) \\ &= (t(s^\sigma)(n), u(s^\sigma)(n)) \\ &= h(s^\sigma)(n). \end{aligned}$$

- os triângulos comutam, isto é, $\pi_1^{\mathcal{N}}(h(s)) = t(s)$ e $\pi_2^{\mathcal{N}}(h(s)) = u(s)$.
  Para todo o $n \in \mathcal{N}$ temos

$$\begin{aligned}\pi_1^{\mathcal{N}}(h(s)(n)) &= \pi_1^{\mathcal{N}}((t(s), u(s))(n)) \\ &= t(s)(n).\end{aligned}$$

  De modo idêntico prova-se que $\pi_2^{\mathcal{N}}(h(s)) = u(s)$.

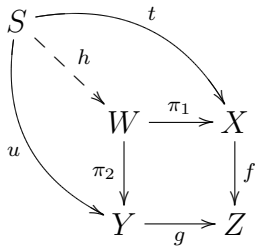
Quanto aos restantes functores que compõem F, vamos apenas apresentar o diagrama e a definição da função h para cada caso, pois a partir daí o resultado mostra-se facilmente.
Para o functor constante $\mathcal{N}$ vem



onde $1_{\mathcal{N}} : \mathcal{N} \to \mathcal{N}$ é a função identidade. Neste caso, tomamos

$$h(s) = t(s)\,(= u(s)).$$

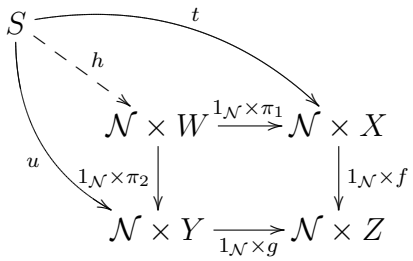
Para functor identidade, Id, tem-se



para

$$h(s) = (t(s), u(s)).$$

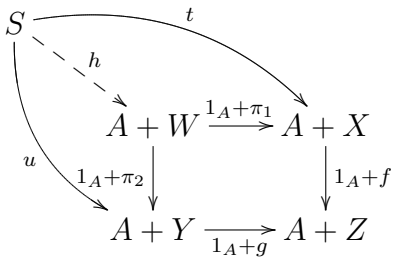
Para o produto cartesiano $\mathcal{N} \times \_$, vem



com

$$h(s) = (t(s), u(s)).$$

Para a soma $A + \_$, vem



com

$$h(s) = \begin{cases} (0, a), & \text{se } t(s) = (0, a) = u(s); \\ (1, (x, y)), & \text{se } t(s) = (1, x) \text{ e } u(s) = (1, y). \end{cases}$$

□

Fica concluída a demonstração do seguinte resultado:

Proposição 2.3.14 *O functor*

$$F = \mathcal{P}_{fin}(Id + (\mathcal{N} \times \mathcal{N} \times Id) + (\mathcal{N} \times Id^{\mathcal{N}}) + (\mathcal{N} \times Id^{\mathcal{N}}))$$

satisfaz as seguintes condições, para todo o f morfismo de coálgebras-F:

(i) *O núcleo de equivalência de f, $E_f = \{(s,t) : f(s) = f(t)\}$, é uma bissimulação.*

(ii) *A imagem* $\{(f(s), f(t)) : (s,t) \in R\}$ *de uma bissimulação* R *por* f *é uma F-bissimulação.*

Demonstração Resulta das proposições 2.3.11, 2.3.12 e 2.3.13. □

Capítulo 3

Semântica de uma linguagem com sincronização restrita

Neste capítulo vamos explorar duas técnicas distintas para estudar a semântica de uma linguagem com uma estrutura simples mas que ilustra bem alguns dos aspectos presentes nas linguagens concorrentes mais comuns. As técnicas que vamos utilizar são a teoria dos *conjuntos com famílias de equivalência*, abreviadamente cfe's, [Mon98], e a representação coalgébrica de sistemas, [Rut00]. O estudo será feito sobre uma linguagem apresentada por J. De Bakker e E. De Vink em [BV96] com a designação $\mathcal{L}_{syn}$.

O capítulo está dividido em quatro partes. Na primeira apresentamos a linguagem $\mathcal{L}_{syn}$, na segunda vamos utilizar a teoria dos conjuntos com famílias de equivalência para a definição das semânticas operacional e denotacional da linguagem $\mathcal{L}_{syn}$. Na terceira parte as semânticas são definidas à custa da existência de coálgebras finais. Na quarta e última parte faz-se um estudo comparativo dos resultados obtidos na segunda e na terceira parte.

3.1 A sintaxe de $\mathcal{L}_{syn}$ e definições básicas

A linguagem $\mathcal{L}_{syn}$ tem uma estrutura simples mas ilustra bem diversos aspectos presentes nas linguagens mais comuns, como sejam:

- *recursividade*, a variável x corresponde a uma chamada ao procedimento associado a x (na declaração).

- *composição sequencial*, sintacticamente representada por $s_1; s_2$ e que corresponde à execução de s_1 seguida da execução de s_2.

- *escolha não determinista*, representada sintacticamente por $s_1 + s_2$ e que corresponde à escolha arbitária entre os operandos s_1 e s_2 e prossegue com a execução do operando escolhido.

- *composição paralela com sincronização* que é expressa sintacticamente (e semanticamente) pelo operador $\|$. Numa primeira aproximação podemos pensar na *composição paralela* $s_1 \|' s_2$ que corresponde ao intercalamento arbitrário das acções atómicas que resultam da execução de s_1 e s_2. Por outras palavras, $s_1 \|' s_2$ irá resultar no conjunto de todas as sequências formadas por uma mistura (fusão) das acções sucessivas de s_1 e s_2 desde que respeitem a ordem pela qual estas acções surgem em s_1 e s_2. A composição paralela com sincronização é um refinamento desta definição em que é possível inserir um ponto de sincronização entre dois comandos a executarem em composição paralela, assim, temos uma execução em paralelo em que o agente que atingir primeiro o ponto de sincronização suspende a sua actividade até que o outro agente atinja, também, o respectivo ponto de sincronização e quando isto acontece voltam a executar novamente os dois em paralelo.

- *restrição*, escreve-se $s \setminus c$ e lê-se "s restringido por c", e corresponde à execução de s excepto no que envolve a acção c e a sua complementar, $\overline{c}$, que não podem ser executadas e portanto a sua ocorrência, sem

a existência de uma acção alternativa, dá origem a uma paragem na execução.

Em linguagens mais refinadas o conceito de ponto de sincronização pode ser utilizado para vários fins como por exemplo a transmissão de informação, mas aqui, iremos limitar-nos à sincronização das duas execuções. A noção de restrição também assume aqui contornos muito elementares mas em linguagens mais complexas pode ser estendida de diversas formas, nomeadamente para modelar fenómenos de mobilidade, conforme iremos ver no capítulo 4.

A notação $(v \in)C$ será utilizada para referirmos o conjunto C e, simultaneamente, a variável v que toma valores em C.

Os programas da linguagem $\mathcal{L}_{syn}$ são constituídos por duas componentes as *instruções* e as *declarações*. As declarações $(D \in)Decl$ podem ser encaradas como funções que atribuem a cada variável de procedimento um conjunto de instruções com características especiais. As instruções $(s \in)Stat$ são construídas a partir das componentes básicas *acções* e *variáveis de procedimento*.

As acções podem ser de dois tipos: acções internas, que podem ser vistas como operações abstractas cuja interpretação não nos interessa aprofundar; ou acções de sincronização.

Seja $(b \in)IAct$ o conjunto das acções internas, e seja τ um elemento especifico de $IAct$ que denota uma acção especial que iremos designar por acção *muda*. Seja $(c \in)Sync$ o conjunto das acções de sincronização que assumimos emparelhadas, isto é a cada acção $c \in Sync$ está associada uma acção complementar $\overline{c}$ que por sua vez tem como complementar c. Matemáticamente, assumimos uma aplicação $\bar{\cdot} : Sync \rightarrow Sync$ tal que para todo o c tem-se $\overline{\overline{c}} = c$. Seja $(a \in)Act = IAct \cup Sync$.

Os elementos do conjunto das variáveis de procedimento $(x \in) PVar$ estão associados a declarações, que podem ser encaradas como partes de programa, e lhes atribuem significado. As declarações fazem parte do programa. A ocorrência de uma variável de procedimento, x, numa instrução – uma chamada de x – acarreta a execução das instruções associadas a x na

declaração. Qualquer dos dois conjuntos Act e $PVar$ poderá ser finito ou infinito. Necessitamos também de introduzir o conjunto $(g \in)GStat$ (subconjunto do conjunto $Stat$) que é o conjunto das instruções que começam por uma acção que "guarda" as variáveis de procedimento que poderão ocorrer na mesma instrução e por isso se designa conjunto das instruções guardadas.

Na especificação da sintaxe de $\mathcal{L}_{syn}$ iremos seguir o formato BNF, que também é utilizado em [BV96], donde se salienta a utilização do símbolo "::="com o significado "é definido(a) como" e o símbolo "|"com o significado "ou". As entidades que ocorrem no lado direito das definições sintácticas são ou dadas a priori (caso dos símbolos '(', ')' ou ';' e $a \in Act$ ou $x \in PVar$) ou são ocorrências recursivas das entidades que estamos a definir.

Definição 3.1.1 (Programas de $\mathcal{L}_{syn}$) Sejam $(a \in)Act$, $(x \in)PVar$ e $(c \in)Sync$ conjuntos dados.

(a) O conjunto $(s \in)Stat$ é dado por

$$s ::= a \mid x \mid (s; s) \mid (s + s) \mid (s \parallel s) \mid s \setminus c$$

(b) O conjunto $(g \in)GStat$ é definido por

$$g ::= a \mid (g; s) \mid (g + g) \mid (g \parallel g) \mid g \setminus c$$

(c) O conjunto das declarações $(D \in)Decl$ é definido por

$$Decl = PVar \rightarrow GStat$$

(d) O conjunto dos programas $(\pi \in)\mathcal{L}_{syn}$ é definido por

$$\mathcal{L}_{syn} = Decl \times Stat$$

□

Para aliviar a notação iremos frequentemente suprimir as referências às declarações D nas situações em que tal informação não seja relevante.

Antes de apresentarmos o sistema de transições para $\mathcal{L}_{syn}$, temos necessidade de introduzir uma nova classe sintáctica designada por reposições.

Definição 3.1.2 (Reposições) O conjunto $(r \in)Res$ das *reposições* é definido por

$$r ::= E|s$$

onde E é um símbolo especial que denota a terminação ou instrução vazia.

□

Para cada $s \in Stat$ convencionaremos que se tem $E; s = s \parallel E = E \parallel s = s$ e $E \parallel E = E \setminus c = E$.

Finalmente vamos apresentar a especificação do sistema de transições $\mathcal{T}_{\mathcal{L}_{syn}}$ que será utilizado para definir a semântica de $\mathcal{L}_{syn}$.

Definição 3.1.3 (Sistema de transições para $\mathcal{L}_{syn}$) $\mathcal{T}_{syn}$=(Decl × Res, Act, → , Spec). As transições de $\mathcal{T}_{syn}$, descritas pela relação →, são tuplos da forma $((D_1|r_1), a, (D_2|r_2))$. Como o nosso estudo contempla apenas os casos em que $D_1 = D_2$ iremos omitir as referências às declarações e utilizar a notação simplificada $r_1 \xrightarrow{a}_D r_2$, onde D=$D_1 = D_2$.

Os axiomas e regras de Spec que caracterizam → são

- (Act) $a \xrightarrow{a}_D E$
- (Rec) $\dfrac{g \xrightarrow{a}_D r}{x \xrightarrow{a}_D r}$ se $D(x) = g$
- (Seq) $\dfrac{s_1 \xrightarrow{a}_D r_1}{s_1; s_2 \xrightarrow{a}_D r_1; s_2}$
- (Choice) $\dfrac{s_1 \xrightarrow{a}_D r}{\begin{array}{c} s_1 + s_2 \xrightarrow{a}_D r \\ s_2 + s_1 \xrightarrow{a}_D r \end{array}}$
- (Par) $\dfrac{s_1 \xrightarrow{a}_D r}{\begin{array}{c} s_1 \parallel s_2 \xrightarrow{a}_D r \parallel s_2 \\ s_2 \parallel s_1 \xrightarrow{a}_D s_2 \parallel r \end{array}}$

- (Sync) $\dfrac{s_1 \xrightarrow{c}_D r_1 \qquad s_2 \xrightarrow{\overline{c}}_D r_2}{s_1 \parallel s_2 \xrightarrow{\tau}_D r_1 \parallel r_2}$

- (Restr) $\dfrac{s \xrightarrow{a}_D r}{s \setminus c \xrightarrow{a}_D r \setminus c} \quad a \neq c, \overline{c}$

□

A intuição operacional das regras é a seguinte:

O axioma (Act) estabelece que uma acção a pode efectuar uma transição para E (e termina) produzindo como acção observável a.

A regra (Rec) indica que as transições de uma variável de procedimento x são exactamente as transições possíveis para g, o conjunto de instruções associado a x na declaração D.

A regra (Seq) cobre duas situações dependendo de r_1 ser a terminação, E, ou ser diferente de E. Assim, se s_1 termina após uma transição com a tem-se que $s_1; s_2$ efectua uma transição com a para s_2, atendendo à convenção $E; s_2 = s_2$. Caso s_1 tenha uma transição com a para s_1' então $s_1; s_2$ transita com etiqueta a para $s_1'; s_2$.

A regra (Choice), com uma premissa e duas conclusões, é uma forma abreviada de estabelecer que numa escolha não determinista, $s_1 + s_2$, se uma das componentes s_1 ou s_2 tem uma transição com etiqueta a para r então $s_1 + s_2$ também tem uma transição com a para r.

A regra (Par) estabelece que a partir da transição $s_1 \xrightarrow{a}_D r$ podemos inferir que uma instrução de composição paralela que envolva s_1 e outra instrução s_2, tem uma transição com a para outra instrução de composição paralela semelhante, mas onde s_1 dá lugar a r. Naturalmente que se $r = E$ a transição é para s_2, atendendo à convenção $s \parallel E = E \parallel s = s$.

(Sync) é a regra para a sincronização: se s_1 pode efectuar uma transição com c que tem como resultado r_1 e s_2 pode efectuar uma transição com $\overline{c}$ para r_2, então $s_1 \parallel s_2$ pode efectuar uma transição com etiqueta τ da qual resulta $r_1 \parallel r_2$. De salientar que esta regra é simétrica uma vez que $\overline{\overline{c}} = c$ e que, além disso, $s_1 \parallel s_2$ pode executar apenas a transição associada a s_1 ou a s_2 com base na regra (Par). Assim, nestas condições, $s_1 \parallel s_2$ pode prosseguir com um de entre os seguintes passos:

(i) executar somente o passo associado à transição de s_1;

(ii) executar somente o passo associado à transição de s_2;

(iii) efectuar um passo de sincronização (que afecta ambos os operandos).

A regra (Restr) estabelece que $s \setminus c$ pode efectuar todas as transições de s em que a acção observável é diferente de c e de $\overline{c}$. Caso s apenas possa efectuar transições com c ou $\overline{c}$, então $s \setminus c$ não tem qualquer transição.

Definição 3.1.4 (Bloqueia) Dizemos que s *bloqueia* se não existem b ($\in IAct$) e r ($\in Res$) tais que $s \xrightarrow{b}_D r$. □

De acordo com o sistema de transições o bloqueio poderá acontecer por dois motivos: ou porque numa instrução de restrição, $s \setminus c$, todas as alternativas para prosseguir com a execução de s iniciarem com c ou $\overline{c}$; ou porque, dada uma acção de sincronização c, não é possível a evolução do sistema de forma a permitir a execução da acção complementar $\overline{c}$ de forma a poder ser aplicável a regra *Sync*.

3.2 Semântica utilizando Conjuntos com Famílias de Equivalência

Nesta secção vamos utilizar a teoria dos cfe's para a definição das semânticas operacional e denotacional de uma linguagem com sincronização restrita, utilizando como domínio a solução de uma equação de domínios sobre cfe's.

3.2.1 Semântica Operacional

Vamos definir a semântica operacional de $\mathcal{L}_{syn}$ tomando como codomínio a solução de uma equação de domínios sobre o functor

$$F(Q) = \{p_\varepsilon\} + \mathcal{P}_{co}(IAct \times Q^\circ)$$

onde $\{p_\varepsilon\}$ e $IAct$ são cfe's discretos dados, e os operadores $+$, $\times$, $.^\circ$ e $\mathcal{P}_{co}(.)$ são os functores definidos na secção 2.1.

A semântica operacional, denotada por $\mathcal{O}$, tem como domínio $Decl \times Stat$ e como codomínio $(p \in)\mathbb{P}_\mathcal{O}$, o cfe solução (única, conforme se viu na secção 2.1) da equação de domínios $X = F(X)$, isto é

$$\mathbb{P}_\mathcal{O} \cong \{p_\varepsilon\} + \mathcal{P}_{co}(IAct \times \mathbb{P}_\mathcal{O}^\circ)$$

Para tornar a escrita mais leve e os raciocínios mais claros, em alguns casos, vamos tratar a equação como uma igualdade sem nos referirmos explicitamente ao isomorfismo que lhe está subjacente. Assim, um processo $p \in \mathbb{P}_\mathcal{O}$ é (representado por): ou o processo nulo p_ε ou um conjunto compacto (possivelmente vazio) de pares $< b_1, p_1 >, ..., < b_i, p_i >, ...$, onde, para cada $i = 1, 2, ...$, tem-se $b_i \in IAct$ e p_i um elemento de $\mathbb{P}_\mathcal{O}$.

A semântica operacional de $\mathcal{L}_{syn}$ é obtida com base numa função auxiliar $\mathcal{O}_0 : Decl \times Res \to \mathbb{P}_\mathcal{O}$.

Para aliviar a notação, sempre que não houver perigo de confusão iremos omitir a referência ao conjunto $(D \in)Decl$.

Vamos então começar por definir $\mathcal{O}_0$.

Definição 3.2.1 (Base da semântica operacional) Seja $\mathcal{O}_0 : Res \to \mathbb{P}_{\mathcal{O}}$ dada por:

- $\mathcal{O}_0(E) = \vec{p_\varepsilon}$ onde $\vec{p_\varepsilon}$ é a sucessão $(\vec{p_\varepsilon}\,[n])_{n\geq 0}$ definida por
  $\vec{p_\varepsilon}[0] = \emptyset$
  $\vec{p_\varepsilon}[n+1] = p_\varepsilon$
- $\mathcal{O}_0(s) = (\mathcal{O}_0(s)[n])_{n\geq 0}$ onde

  $\mathcal{O}_0(s)[0] = \emptyset$
  $\mathcal{O}_0(s)[n+1] = \{(b, \mathcal{O}_0(r)[n]) : s \xrightarrow{b}_D r\}$

□

Da definição apercebemo-nos que p_ε expressa a terminação normal enquanto que $\emptyset$ expressa a terminação por bloqueio. Repare-se que apenas as acções internas ($b \in IAct$) contribuem para o resultado. Uma consequência desse facto é que os passos com etiqueta c para os quais não existir o passo complementar, que podem ser vistos como tentativas de sincronização falhadas, não deixam qualquer rasto no resultado excepto se não for possível executar um passo com uma acção interna e, nesse caso, há um bloqueio.

Para todo $n \geq 0$, defina-se $\mathbb{P}_n = F^n(\{\emptyset\})$. Para mostrar que $\mathcal{O}_0$ está bem definida temos de mostrar que:

Lema 3.2.2

(a) $\mathcal{O}_0(r)[n] \in \mathbb{P}_n, \forall_n \geq 0$

(b) $\beta_n(\mathcal{O}_0(r)[n+1]) = \mathcal{O}_0(r)[n]$.

Demonstração As demonstrações são feitas por indução em n.

(a) O caso $n = 0$ é imediato. Para $n+1$, se $r = E$ vem $\mathcal{O}_0(E)[n+1] = p_\varepsilon$ que pertence a $\mathbb{P}_{n+1}$. Se $r = s$, vem $\mathcal{O}_0(s)[n+1] = \{(b, \mathcal{O}_0(r)[n]) : s \xrightarrow{b}_D r\}$ onde $\mathcal{O}_0(r)[n] \in \mathbb{P}_n$ por hipótese de indução e logo $\mathcal{O}_0(s)[n+1] \in \mathbb{P}_{n+1}$, como se pretendia demonstrar.

(b) Para $n = 0$ é imediato que, pela definição de α, se tem $\beta_0(\mathcal{O}_0(E)[1]) = \emptyset = \mathcal{O}_0(E)[0]$ e $\beta_0(\mathcal{O}_0(s)[1]) = \emptyset = \mathcal{O}_0(s)[0]$. Suponhamos verdadeira a igualdade $\beta_n(\mathcal{O}_0(r)[n+1]) = \mathcal{O}_0(r)[n]$, vamos demonstrar, por indução na estrutura de r que $\beta_{n+1}(\mathcal{O}_0(r)[n+2]) = \mathcal{O}_0(r)[n+1]$:

Se $r = E$ tem-se

$\beta_{n+1}(\mathcal{O}_0(E)[n+2]) = \beta_{n+1}(p_\varepsilon) = p_\varepsilon = \mathcal{O}_0(E)[n+1]$.

Se $r = s$ temos dois casos a considerar. No caso em que s bloqueia vem

$\beta_{n+1}(\mathcal{O}_0(s)[n+2]) \stackrel{def.\mathcal{O}_0}{=} \beta_{n+1}(\emptyset) = \emptyset = \mathcal{O}_0(s)[n+1]$.

No caso em que s não bloqueia, vem

$$\begin{array}{rcl} \beta_{n+1}(\mathcal{O}_0(s)[n+2]) & \stackrel{def.\mathcal{O}_0}{=} & \beta_{n+1}(\{(b, \mathcal{O}_0(r)[n+1]) : s \stackrel{b}{\rightarrow}_D r\}) \\ & \stackrel{def.\beta_n}{=} & \{(b, \beta_n(\mathcal{O}_0(r)[n+1])) : s \stackrel{b}{\rightarrow}_D r\} \\ & \stackrel{hip.ind.}{=} & \{(b, \mathcal{O}_0(r)[n]) : s \stackrel{b}{\rightarrow}_D r\} \\ & = & \mathcal{O}_0(s)[n+1]. \end{array}$$

□

Definição 3.2.3 (Semântica operacional) A semântica operacional

$$\mathcal{O} : \mathcal{L}_{syn} \rightarrow \mathbb{P}_{\mathcal{O}}$$

é dada por

$$\mathcal{O}(s) = \mathcal{O}_0(s).$$

□

Vamos em seguida apresentar dois exemplos do cálculo da semântica operacional. O primeiro é um caso muito simples e o segundo envolve uma sincronização.

Exemplos 3.2.4

1. $\mathcal{O}((b_1; c) \parallel b_2) = (\mathcal{O}_0((b_1; c) \parallel b_2)[n])_{n \geq 0}$, com

$$
\begin{aligned}
\mathcal{O}_0((b_1;c) \parallel b_2)[0] &= \emptyset \\
\\
\mathcal{O}_0((b_1;c) \parallel b_2)[1] &= \{(b_1, \mathcal{O}_0(c \parallel b_2)[0]), (b_2, \mathcal{O}_0(b_1;c)[0]))\} \\
&= \{(b_1, \emptyset), (b_2, \emptyset)\} \\
\mathcal{O}_0((b_1;c) \parallel b_2)[2] &= \{(b_1, \mathcal{O}_0(c \parallel b_2)[1]), (b_2, \mathcal{O}_0(b_1;c)[1]))\} \\
&= \{(b_1, \{(b_2, \mathcal{O}_0(c)[0])\}), (b_2, \{(b_1, \mathcal{O}_0(c)[0])\})\} \\
&= \{(b_1, \{(b_2, \emptyset)\}), (b_2, \{(b_1, \emptyset)\})\} \\
\mathcal{O}_0((b_1;c) \parallel b_2)[3] &= \{(b_1, \mathcal{O}_0(c \parallel b_2)[2]), (b_2, \mathcal{O}_0(b_1;c)[2]))\} \\
&= \{(b_1, \{(b_2, \mathcal{O}_0(c)[1])\}), (b_2, \{(b_1, \mathcal{O}_0(c)[1])\})\} \\
&= \{(b_1, \{(b_2, \emptyset)\}), (b_2, \{(b_1, \emptyset)\})\} \\
\mathcal{O}_0((b_1;c) \parallel b_2)[3+i] &= \{(b_1, \{(b_2, \emptyset)\}), (b_2, \{(b_1, \emptyset)\})\}, i = 1, 2, ...
\end{aligned}
$$

2. $\mathcal{O}((b_1;c) \parallel (b_2;\overline{c})) = (\mathcal{O}_0((b_1;c) \parallel (b_2;\overline{c}))[n])_{n \geq 0}$, com

$$
\begin{aligned}
\mathcal{O}_0((b_1;c) \parallel (b_2;\overline{c}))[0] &= \emptyset \\
\mathcal{O}_0((b_1;c) \parallel (b_2;\overline{c}))[1] &= \{(b_1, \mathcal{O}_0(c \parallel (b_2;\overline{c}))[0]), (b_2, \mathcal{O}_0((b_1;c) \parallel \overline{c})[0])\} \\
&= \{(b_1, \emptyset), (b_2, \emptyset)\} \\
\mathcal{O}_0((b_1;c) \parallel (b_2;\overline{c}))[2] &= \{(b_1, \mathcal{O}_0(c \parallel (b_2;\overline{c}))[1]), (b_2, \mathcal{O}_0((b_1;c) \parallel \overline{c})[1])\} \\
&= \{(b_1, \{(b_2, \emptyset)\}), (b_2, \{(b_1, \emptyset)\})\} \\
\mathcal{O}_0((b_1;c) \parallel (b_2;\overline{c}))[3] &= \{(b_1, \mathcal{O}_0(c \parallel (b_2;\overline{c}))[2]), (b_2, \mathcal{O}_0((b_1;c) \parallel \overline{c})[2])\} \\
&= \{(b_1, \{(b_2, \mathcal{O}_0(c \parallel \overline{c})[1])\}), (b_2, \{(b_1, \mathcal{O}_0(c \parallel \overline{c})[1])\})\} \\
&= \{(b_1, \{(b_2, \{(\tau, \mathcal{O}_0(E)[0])\}\}), \\
&\qquad\qquad (b_2, \{(b_1, \{(\tau, \mathcal{O}_0(E)[0])\})\})\} \\
&= \{(b_1, \{(b_2, \{(\tau, \emptyset)\})\}), (b_2, \{(b_1, \{(\tau, \emptyset)\})\})\} \\
\mathcal{O}_0((b_1;c) \parallel (b_2;\overline{c}))[4] &= \{(b_1, \mathcal{O}_0(c \parallel (b_2;\overline{c}))[3]), (b_2, \mathcal{O}_0((b_1;c) \parallel \overline{c})[3])\} \\
&= \{(b_1, \{(b_2, \mathcal{O}_0(c \parallel \overline{c})[2])\}), (b_2, \{(b_1, \mathcal{O}_0(c \parallel \overline{c})[2])\})\} \\
&= \{(b_1, \{(b_2, \{(\tau, \mathcal{O}_0(E)[1])\})\}), \\
&\qquad\qquad (b_2, \{(b_1, \{(\tau, \mathcal{O}_0(E)[1])\})\})\} \\
&= \{(b_1, \{(b_2, \{(\tau, p_\varepsilon)\})\}), (b_2, \{(b_1, \{(\tau, p_\varepsilon)\})\})\}
\end{aligned}
$$

e para i=1,2,... vem

$$\mathcal{O}_0((b_1; c) \parallel (b_2; \overline{c}))[i+4] \quad = \quad \{(b_1, \{(b_2, \{(\tau, p_\varepsilon)\})\}), (b_2, \{(b_1, \{(\tau, p_\varepsilon)\})\})\}.$$

□

3.2.2 Semântica Denotacional

Vamos agora definir a semântica denotacional de $\mathcal{L}_{syn}$. O domínio semântico, $\mathbb{P}_{\mathcal{D}}$, que vamos utilizar nesta secção é ligeiramente diferente do utilizado na definição da semântica operacional. A diferença deve-se ao facto que, para que a semântica seja composicional, na sua construção temos de ter em conta também as acções de sincronização, pelo que no cálculo de $\mathbb{P}_{\mathcal{D}}$ vamos utilizar o conjunto $Act = IAct \cup Sync$ onde no domínio da semântica operacional utilizávamos $IAct$. Assim, vamos definir $\mathcal{D} : Decl \times Res \to \mathbb{P}_{\mathcal{D}}$ onde $(p \in)\mathbb{P}_{\mathcal{D}}$ é o cfe solução (única) da equação de domínios

$$\mathbb{P}_{\mathcal{D}} \cong \{p_\varepsilon\} + \mathcal{P}_{co}(Act \times \mathbb{P}^0_{\mathcal{D}})$$

Tal como no caso do domínio da semântica operacional, vamos tratar a equação como uma igualdade sem nos referirmos explicitamente ao isomorfismo. Nesta secção, sempre que não houver perigo de confusão, para aliviar a notação, iremos referir-nos a $\mathbb{P}_{\mathcal{D}}$ como apenas $\mathbb{P}$.

Antes de introduzirmos a definição da semântica denotacional propriamente dita, é necessário definirmos alguns operadores auxiliares e introduzirmos um mecanismo para a atribuição de significado no domínio semântico às variáveis de procedimento. Para facilitar a escrita, iremos frequentemente referir-nos a $p[n]$, n-ésima componente de p, simplesmente por p_n. Passemos então à definição dos operadores auxiliares.

Definição 3.2.5 (Operadores auxiliares)

(a) $\oplus : \mathbb{P} \times \mathbb{P} \to \mathbb{P}$ é definido por

$(p \oplus q) = (p_n \oplus_n q_n)_{n \geq 0}$, onde o operador $\oplus_n : \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$ é definido por

$$\emptyset \oplus_0 \emptyset = \emptyset$$

$$u \oplus_{n+1} v = \begin{cases} v & \text{se} \quad u = p_\varepsilon, \\ u & \text{se} \quad v = p_\varepsilon, \\ u \cup v & \text{se } u \neq p_\varepsilon \text{ e } v \neq p_\varepsilon. \end{cases}$$

(b) $\odot : \mathbb{P} \times \mathbb{P} \to \mathbb{P}$ é definido por

$(p \odot q) = (p_n \odot_n q_n)_{n \geq 0}$ onde o operador $\odot_n : \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$ é definido por

$$\emptyset \odot_0 \emptyset = \emptyset$$

$$u \odot_{n+1} v = \begin{cases} v & \text{se} \quad u = p_\varepsilon, \\ \{(a, x \odot_n \beta_n(v)) : (a, x) \in u\} & \text{caso contrário.} \end{cases}$$

(c) $\between : \mathbb{P} \times \mathbb{P} \to \mathbb{P}$ é definido por

$(p \between q) = (p_n \between_n q_n)_{n \geq 0}$

onde o operador $\between_n : \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$ é definido por

$$\emptyset \between_0 \emptyset = \emptyset$$

$$u \between_{n+1} v = u|\!\lfloor_{n+1} v \ \cup \ v|\!\lfloor_{n+1} u \ \cup \ u \mid_{n+1} v$$

onde

$$u|\!\lfloor_{n+1} v = \begin{cases} v & \text{se} \quad u = p_\varepsilon, \\ \{(a, x \between_n \beta_n(v)) : (a, x) \in u\} & \text{caso contrário;} \end{cases}$$

e

$u \mid_{n+1} v = \cup\{(\tau, x \between_n y) : (c, x) \in u, (\overline{c}, y) \in v\}$.

(d) $\leftthreetimes : \mathbb{P} \times Sync \to \mathbb{P}$ é definido por

$(p \leftthreetimes c) = (p_n \leftthreetimes_n c)_{n \geq 0}$

onde o operador $\leftthreetimes_n : \mathbb{P}_n \times Sync \to \mathbb{P}_n$ é definido por

$$\emptyset \diagdown\!\!\diagdown_0 c = \emptyset$$

$$u \diagdown\!\!\diagdown_{n+1} c = \begin{cases} p_\varepsilon & \text{se} \quad u = p_\varepsilon, \\ \{(a, x \diagdown\!\!\diagdown_n c) : (a, x) \in u, a \neq c, a \neq \overline{c}\} & \text{caso contrário.} \end{cases}$$

□

Estes operadores $\oplus, \odot, \otimes$ e $\diagdown\!\!\diagdown$ irão ser os correspondentes semânticos dos operadores sintácticos $+$, $;$, $\|$ e $\backslash$, respectivamente.

Lema 3.2.6 *Os operadores* $\oplus, \odot, \otimes$ *e* $\diagdown\!\!\diagdown$

1. *estão bem definidos;*
2. *são conservadores.*

Demonstração

1. As provas de que os operadores estão bem definidos são feitas por indução, vamos ver em pormenor os casos do primeiro e segundo operador, os restantes são semelhantes.

- $\oplus$ está bem definido?

  Ora se $p, q \in \mathbb{P}$ tem-se $p_n, q_n \in \mathbb{P}_n$ para todo $n \geq 0$, logo $p_n \oplus_n q_n \in \mathbb{P}_n$, pelo que $(p \oplus q)_n \in \mathbb{P}_n$ para todo n. Por outro lado, da definição de $\oplus$ temos

  $\beta_0((p \oplus q)_1) = \emptyset = (p \oplus q)_0$

$$\beta_{n+1}((p\oplus q)_{n+2}) \overset{def.\oplus}{=} \begin{cases} \beta_{n+1}(q_{n+2}) & \text{se} \quad p_{n+2} = p_\varepsilon, \\ \beta_{n+1}(p_{n+2}) & \text{se} \quad q_{n+2} = p_\varepsilon, \\ \beta_{n+1}(p_{n+2} \cup q_{n+2}) & \text{nos restantes casos.} \end{cases}$$

$$\overset{def.\beta_n}{=} \begin{cases} \beta_{n+1}(q_{n+2}) & \text{se} \quad p_{n+2} = p_\varepsilon, \\ \beta_{n+1}(p_{n+2}) & \text{se} \quad q_{n+2} = p_\varepsilon, \\ \beta_{n+1}(p_{n+2}) \cup \beta_{n+1}(q_{n+2}) & \text{nos restantes casos.} \end{cases}$$

$$\overset{def.\beta_n \& Nota}{=} \begin{cases} q_{n+1} & \text{se} \quad p_{n+1} = p_\varepsilon, \\ p_{n+1} & \text{se} \quad q_{n+1} = p_\varepsilon, \\ p_{n+1} \cup q_{n+1} & \text{nos restantes casos.} \end{cases}$$

$$= \quad (p\oplus q)_{n+1}.$$

- $\dot{\cup}$ está bem definido?

  Em primeiro lugar note-se que se $u, v \in \mathbb{P}_n$ mostra-se por indução em n que $u\dot{\cup}_n v \in \mathbb{P}_n$. Resta mostrar que $\beta_n((p\dot{\cup}q)_{n+1}) = (p\dot{\cup}q)_n$. Vamos proceder à demonstração por indução em n.

  Para $n = 0$ vem

  $$\beta_0((p\dot{\cup}q)_1) = \emptyset = (p\dot{\cup}q)_0.$$

  Suponhamos a afirmação verdadeira para n, isto é,

  $$\beta_n((p\dot{\cup}q)_{n+1}) = (p\dot{\cup}q)_n$$

  que é o mesmo que,

  $$\beta_n(p_{n+1}\dot{\cup}_{n+1}q_{n+1}) = p_n\dot{\cup}_n q_n.$$

  Temos de mostrar que

  $$\beta_{n+1}((p\dot{\cup}q)_{n+2}) = (p\dot{\cup}q)_{n+1}$$

ou seja, que

$$\beta_{n+1}((p \dot{;} q)_{n+2}) = p_{n+1} \dot{;}_{n+1} q_{n+1}.$$

Ora

$\beta_{n+1}((p \dot{;} q)_{n+2}) = \beta_{n+1}(p_{n+2} \dot{;}_{n+2} q_{n+2})$

$$\overset{def.\dot{;}_n}{=} \begin{cases} \beta_{n+1}(q_{n+2}) & \text{se} \quad p_{n+2} = p_\varepsilon, \\ \beta_{n+1}(\cup\{(b, x \dot{;}_{n+1} \beta_{n+1}(q_{n+2})) : (b,x) \in p_{n+2}\}) & \text{caso contrário.} \end{cases}$$

$$\overset{def.\beta_n}{=} \begin{cases} q_{n+1} & \text{se} \quad p_{n+2} = p_\varepsilon, \\ \cup\{(b, \beta_n(x \dot{;}_{n+1} q_{n+1})) : (b,x) \in p_{n+2}\} & \text{caso contrário.} \end{cases}$$

$$\overset{h.ind.\&Nota}{=} \begin{cases} q_{n+1} & \text{se} \quad p_{n+1} = p_\varepsilon, \\ \{(b, \beta_n(x) \dot{;}_n \beta_n(q_{n+1})) : (b, \beta_n(x)) \in p_{n+1}\} & \text{caso contrário.} \end{cases}$$

$= (p_{n+1} \dot{;}_{n+1} q_{n+1})$, porque β_n é sobrejectiva.

Logo se $p, q \in \mathbb{P}$ então $p \dot{;} q \in \mathbb{P}$

Nota: Em algumas das igualdades usamos também o facto que $p_{n+2} = p_\varepsilon$ se e só se $p_{n+1} = p_\varepsilon$, visto que,

$p_{n+2} = p_\varepsilon \Rightarrow \beta_n(p_{n+2}) = p_\varepsilon = p_{n+1}$, e

$p_{n+1} = p_\varepsilon \Rightarrow \beta_n(p_{n+2}) = p_\varepsilon \Rightarrow p_{n+2} = p_\varepsilon$, porque o único elemento aplicado em p_ε é o próprio p_ε.

2. Para verificar que os operadores são conservadores vamos ver em pormenor apenas o caso de $\dot{;}$, pois os outros casos são semelhantes.

$\dot{;}$ é conservador?

O caso $n = 0$ é trivial. Vamos verificar se, para $n \geq 1$, se tem

$$\left.\begin{array}{l} p \equiv_n u \\ q \equiv_n v \end{array}\right\} \Rightarrow p \dot{;} q \equiv_n u \dot{;} v$$

Ora, se $p \equiv_n u$ e $q \equiv_n v$, pela definição de $\equiv_n$ vem $p_n = u_n$ e $q_n = v_n$. Logo,

$$(p \odot q)_n = (p_n \odot_n q_n) = (u_n \odot_n v_n) = (u \odot v)_n$$

Donde, novamente pela definição de $\equiv_n$, vem

$$p \odot q \equiv_n u \odot v$$

□

Para a fase que se segue vamos ter de introduzir uma camada intermédia constituída por um conjunto de funções definidas de $PVar$ em $\mathbb{P}$ que serão designadas por ambientes. São estas funções, que serão controladas pelas declarações $D \in Decl$, que vão atribuir às variáveis de procedimento um significado no domínio semântico.

Definição 3.2.7 (Ambientes) O conjunto de *ambientes* denotado por $(\rho \in) Env$ é

$$Env = PVar \to \mathbb{P}.$$

□

Agora estamos em condições de definir as funções auxiliares $\mathcal{D}_1$ e Ψ_D à custa das quais definiremos então a semântica denotacional para $\mathcal{L}_{syn}$.

Definição 3.2.8 (Função denotacional auxiliar)

$\mathcal{D}_1 : Res \to Env \to \mathbb{P}$ é definida por

$$\begin{aligned}
&\mathcal{D}_1(E)(\rho) = \overrightarrow{p_\varepsilon} \\
&\mathcal{D}_1(a)(\rho) = \overrightarrow{a} \\
&\mathcal{D}_1(x)(\rho) = \rho(x) \\
&\mathcal{D}_1(s_1 + s_2)(\rho) = \mathcal{D}_1(s_1)(\rho) \oplus \mathcal{D}_1(s_2)(\rho) \\
&\mathcal{D}_1(s_1; s_2)(\rho) = \mathcal{D}_1(s_1)(\rho) \odot \mathcal{D}_1(s_2)(\rho) \\
&\mathcal{D}_1(s_1 \parallel s_2)(\rho) = \mathcal{D}_1(s_1)(\rho) \between \mathcal{D}_1(s_2)(\rho) \\
&\mathcal{D}_1(s \setminus c)(\rho) = \mathcal{D}_1(s)(\rho) \oslash c
\end{aligned}$$

onde $\vec{a}= (\vec{a}\ [n])_{n\geq 0}$ com

$$(\vec{a})[0] = \emptyset,\ (\vec{a})[1] = \{(a, \emptyset)\},\ (\vec{a})[n+2] = \{(a, p_\varepsilon)\}$$

□

Facilmente se verifica que $\mathcal{D}_1$ está bem definida. Os três primeiros casos são imediatos e os restantes vêm por indução e pela definição dos operadores semânticos.

Como os ambientes são controladas pelas declarações estabelece-se uma correspondência entre as declarações e os ambientes. A uma declaração $D : PVar \to GStat$ fazemos corresponder um ambiente $\rho_D : PVar \to \mathbb{P}$ que é o ponto fixo de uma função que passamos a definir.

Lema 3.2.9 *Seja* $(\rho \in)Env : PVar \to \mathbb{P}$*, e seja* $\Psi_D : Env \to Env$ *definida por* $\Psi_D(\rho)(x) = \mathcal{D}_1(D(x))(\rho)$*. Então*

(a) $\Psi_D(\rho)(x)$ *está bem definida.*

(b) Ψ_D *é aproximante em* ρ*.*

Demonstração

(a) Dado $x \in PVar$ tem-se $D(x) \in GStat \subseteq Res$ e portanto $\mathcal{D}_1(D(x))(\rho)$ é um elemento de $\mathbb{P}$.

(b) Pretende-se mostrar que dados dois ambientes θ e ρ, se $\rho \equiv_n \theta$ então $\Psi_D(\rho) \equiv_{n+1} \Psi_D(\theta)$. Tem-se que

$$\Psi_D(\rho) \equiv_{n+1} \Psi_D(\theta) \text{ se e só se } \forall x \in PVar,\ \Psi_D(\rho)(x) \equiv_{n+1} \Psi_D(\theta)(x)$$

o que é equivalente a

$$\forall x \in PVar,\ \mathcal{D}_1(D(x))(\rho) \equiv_{n+1} \mathcal{D}_1(D(x))(\theta). \qquad (1)$$

Ora, $\rho \equiv_n \theta$ se e só se $\forall x \in PVar, \rho(x) \equiv_n \theta(x)$, logo supondo verdadeira a afirmação

$$(2)\forall x \in PVar, \rho(x) \equiv_n \theta(x) \Rightarrow \begin{cases} \forall s \in Stat, \mathcal{D}_1(s)(\rho) \equiv_n \mathcal{D}_1(s)(\theta) \\ \forall g \in GStat, \mathcal{D}_1(g)(\rho) \equiv_{n+1} \mathcal{D}_1(g)(\theta) \end{cases}$$

tem-se

$$\forall x \in PVar, \rho(x) \equiv_n \theta(x) \Rightarrow \forall x \in PVar, \mathcal{D}_1(D(x))(\rho) \equiv_{n+1} \mathcal{D}_1(D(x))(\theta)$$

o que, atendendo a (1), demonstra o resultado pretendido.

Passemos, agora, à demonstração de (2) que é feita por indução estrutural em s

- Caso $s = a$: $\mathcal{D}_1(a)(\rho) \stackrel{def.\mathcal{D}_1}{=} \overrightarrow{a} \equiv_{n+1} \overrightarrow{a} = \mathcal{D}_1(a)(\theta)$
- Caso $s = x$: $\mathcal{D}_1(x)(\rho) \stackrel{def.\mathcal{D}_1}{=} \rho(x) \stackrel{h.ind.}{\equiv_n} \theta(x) = \mathcal{D}_1(x)(\theta)$
- Caso $s = s_1 + s_2$: como $\oplus$ é conservador e utilizando a hipótese de indução tem-se

$$\begin{array}{rcl} \mathcal{D}_1(s_1 + s_2)(\rho) & \stackrel{def.\mathcal{D}_1}{=} & \mathcal{D}_1(s_1)(\rho) \oplus \mathcal{D}_1(s_2)(\rho) \\ & \equiv_k & \mathcal{D}_1(s_1)(\theta) \oplus \mathcal{D}_1(s_2)(\theta) \\ & = & \mathcal{D}_1(s_1 + s_2)(\theta) \end{array}$$

  com $k = n + 1$ se s_1 e s_2 forem guardados e $k = n$ caso contrário.

- Caso $s = s_1; s_2$: como $\odot$ é conservador e utilizando a hipótese de indução tem-se

$$\begin{array}{rcl} \mathcal{D}_1(s_1; s_2)(\rho) & \stackrel{def.\mathcal{D}_1}{=} & \mathcal{D}_1(s_1)(\rho) \odot \mathcal{D}_1(s_2)(\rho) \\ & \equiv_k & \mathcal{D}_1(s_1)(\theta) \odot \mathcal{D}_1(s_2)(\theta) \\ & = & \mathcal{D}_1(s_1; s_2)(\theta) \end{array}$$

  onde, tal como anteriormente, $k = n + 1$ se s_1 e s_2 forem guardados e $k = n$ caso contrário.

- Caso $s = s_1 \parallel s_2$: do mesmo modo que nos casos anteriores, tem-se

$$\begin{array}{rcl} \mathcal{D}_1(s_1 \parallel s_2)(\rho) & \stackrel{def.\mathcal{D}_1}{=} & \mathcal{D}_1(s_1)(\rho) \textcircled{\shuffle} \mathcal{D}_1(s_2)(\rho) \\ & \equiv_k & \mathcal{D}_1(s_1)(\theta) \textcircled{\shuffle} \mathcal{D}_1(s_2)(\theta) \\ & = & \mathcal{D}_1(s_1 \parallel s_2)(\theta) \end{array}$$

  onde, tal como anteriormente, $k = n + 1$ se s_1 e s_2 forem guardados e $k = n$ caso contrário.

- Caso $s = s_1 \setminus c$:

$$\begin{array}{rcl} \mathcal{D}_1(s_1 \setminus c)(\rho) & \stackrel{def.\mathcal{D}_1}{=} & \mathcal{D}_1(s_1)(\rho) \textcircled{\backslash} c \\ & \equiv_k & \mathcal{D}_1(s_1)(\theta) \textcircled{\backslash} c \\ & = & \mathcal{D}_1(s_1 \setminus c)(\theta) \end{array}$$

  onde, tal como anteriormente, $k = n + 1$ se s_1 e s_2 forem guardados e $k = n$ caso contrário. □

O facto de Ψ_D ser aproximante garante-nos que tem um único ponto fixo, esse ponto fixo, que vamos denotar por ρ_D, é precisamente o ambiente no qual pretendemos calcular a semântica.

Podemos, finalmente, definir a semântica denotacional para $\mathcal{L}_{syn}$

Definição 3.2.10 (Base da semântica denotacional)

$$\mathcal{D}_0 : Res \to \mathbb{P}$$

é definida por

$$\mathcal{D}_0(r) = \mathcal{D}_1(r)(\rho_D)$$

com $\rho_D = fix(\Psi_D)$.

□

Definição 3.2.11 (Semântica denotacional) A semântica denotacional

$$\mathcal{D} : \mathcal{L}_{syn} \to \mathbb{P}$$

é definida por

$$\mathcal{D}(s) = \mathcal{D}_0(s).$$

□

Vamos apresentar agora um exemplo da utilização desta definição para o mesmo programa que utilizamos no segundo exemplo apresentado para o cálculo da semântica operacional:

Exemplo 3.2.12

$$\begin{aligned}\mathcal{D}((b_1; c) \parallel (b_2; \overline{c}))) &= \mathcal{D}_1((b_1; c) \parallel (b_2; \overline{c})))(\rho_D) \\ &= (\mathcal{D}_1(b_1)(\rho_D) \odot \mathcal{D}_1(c)(\rho_D)) \oplus (\mathcal{D}_1(b_2)(\rho_D) \odot \mathcal{D}_1(\overline{c})(\rho_D)) \\ &= (\overrightarrow{b_1} \odot \overrightarrow{c}) \oplus (\overrightarrow{b_2} \odot \overrightarrow{\overline{c}})\end{aligned}$$

onde

$$\begin{aligned}(\overrightarrow{b_1} \odot_0 \overrightarrow{c})_0 &= \emptyset \\ (\overrightarrow{b_1} \odot_1 \overrightarrow{c})_1 &= \{(b_1, \emptyset)\} \\ (\overrightarrow{b_1} \odot_2 \overrightarrow{c})_2 &= \{(b_1, \{(c, \emptyset)\})\} \\ (\overrightarrow{b_1} \odot_{3+i} \overrightarrow{c})_3 + i &= \{(b_1, \{(c, p_\varepsilon)\})\}\end{aligned}$$

$$\begin{aligned}(\overrightarrow{b_2} \odot_0 \overrightarrow{\overline{c}})_0 &= \emptyset \\ (\overrightarrow{b_2} \odot_1 \overrightarrow{\overline{c}})_1 &= \{(b_2, \emptyset)\} \\ (\overrightarrow{b_2} \odot_2 \overrightarrow{\overline{c}})_2 &= \{(b_2, \{(\overline{c}, \emptyset)\})\} \\ (\overrightarrow{b_2} \odot_{3+i} \overrightarrow{\overline{c}})_{3+i} &= \{(b_2, \{(\overline{c}, p_\varepsilon)\})\}\end{aligned}$$

Façamos $p = (\overrightarrow{b_1} \odot \overrightarrow{c})$ e $q = (\overrightarrow{b_2} \odot \overrightarrow{\overline{c}})$.

$$\mathcal{D}((b_1; c) \parallel (b_2; \overline{c}))) = p \oplus q = (p_n \oplus_n q_n)_{n \geq 0}$$

onde

$$
\begin{aligned}
(p_0 \oplus_0 q_0)_0 &= \emptyset \\
(p_1 \oplus_1 q_1)_1 &= \{(b_1, \emptyset), (b_2, \emptyset)\} \\
(p_2 \oplus_2 q_2)_2 &= \{(b_1, \{(c, \emptyset), (b_2, \emptyset)\}), (b_2, \{(\overline{c}, \emptyset), (b_1, \emptyset)\})\} \\
(p_3 \oplus_3 q_3)_3 &= \{(b_1, \{(c, \{(b_2, \emptyset)\}), (b_2, \{(\overline{c}, \emptyset), (c, \emptyset), (\tau, \emptyset)\})\}), \\
&\quad (b_2, \{(\overline{c}, \{(b_1, \emptyset)\}), (b_1, \{(c, \emptyset), (\overline{c}, \emptyset), (\tau, \emptyset)\})\} \\
&\ldots
\end{aligned}
$$

□

Comparando com o resultado obtido para a semântica operacional a diferença reside essencialmente no facto de as acções de sincronização contribuírem para o resultado, mesmo quando não é efectuada uma sincronização.

3.2.3 Equivalência entre as Semânticas Operacional e Denotacional

Nesta secção vamos mostrar a equivalência entre as duas semânticas definidas para $\mathcal{L}_{syn}$. Pelo facto de a semântica denotacional ter de ser obrigatoriamente composicional tivemos de utilizar para a sua definição um domínio ligeiramente diferente do utilizado para a semântica operacional, por isso não é possível estabelecer a equivalência directa $\mathcal{O} = \mathcal{D}$, uma vez que na semântica operacional não aparecem as acções de sincronização. Em vez disso vamos estabelecer uma relação $\mathcal{O} = abs \circ \mathcal{D}$ onde abs é uma "função de abstracção" que abstrai ("corta") os ramos a partir do ponto em que contiverem c ou $\overline{c}$.

Definição 3.2.13 (Função de abstracção) Seja $abs : \mathbb{P}_{\mathcal{D}} \to \mathbb{P}_{\mathcal{O}}$ definida por

$$abs(p) = (abs_n(p_n))_{n \geq 0}$$

com

$abs_0(u) = \emptyset$

$$abs_{n+1}(u) = \begin{cases} p_\varepsilon & \text{se} \quad u = p_\varepsilon, \\ \{(b, abs_n(x)) : (b, x) \in u, b \in IAct\} & \text{caso contrário.} \end{cases}$$

□

Como não é possível efectuar raciocínios indutivos sobre a estrutura dos programas de $\mathcal{L}_{syn}$, porque falham para o caso das variáveis $x \in PVar$, vamos precisar de uma função, denotada por peso, que atribui um "peso" às reposições. Este peso é um número natural, que é zero para E e é superior a zero no caso das instruções.

Definição 3.2.14 (Função peso) A função $\mathsf{peso} : Decl \times Res \to \mathbb{N}$ é definida por

$$\begin{aligned} \mathsf{peso}(E) &= 0 \\ \mathsf{peso}(a) &= 1 \\ \mathsf{peso}(x) &= \mathsf{peso}(D(x)) + 1 \\ \mathsf{peso}(s_1; s_2) &= \mathsf{peso}(s_1) + 1 \\ \mathsf{peso}(s_1 + s_2) &= max\{\mathsf{peso}(s_1), \mathsf{peso}(s_2)\} + 1 \\ \mathsf{peso}(s_1 \parallel s_2) &= max\{\mathsf{peso}(s_1), \mathsf{peso}(s_2)\} + 1 \\ \mathsf{peso}(s \setminus c) &= \mathsf{peso}(s) + 1 \end{aligned}$$

□

A função está bem definida. Verifica-se primeiro para o subconjunto $GStat$ onde está bem definida por indução na complexidade sintáctica e em seguida verifica-se para o caso geral, novamente por indução sintáctica, onde o único caso que poderia levantar problemas é $\mathsf{peso}(x) = \mathsf{peso}(D(x)) + 1$, mas como $D(x) \in GStat$ tem-se $\mathsf{peso}(D(x))$ bem definido pela primeira parte e logo $\mathsf{peso}(x)$ também está bem definido.

Vamos também necessitar do seguinte resultado:

Lema 3.2.15

$$(a, \mathcal{D}_0(r)) \in \mathcal{D}(s) \text{ se e só se } s \xrightarrow{a}_D r.$$

Demonstração A afirmação é equivalente a

$$(a, \beta_n(\mathcal{D}_1(r)(\rho_D)[n+1]) \in \mathcal{D}_1(s)(\rho_D)[n+1] \text{ se e só se } s \xrightarrow{a}_D r,$$

que é o mesmo que

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s)(\rho_D)[n+1] \text{ se e só se } s \xrightarrow{a}_D r.$$

Vamos então passar à demonstração que será feita por indução em peso.

- Caso $s = a$:

  Atendendo à definição de $\mathcal{T}_{syn}$, a única possibilidade é termos $a \xrightarrow{a}_D E$. Por outro lado, a única possibilidade para $\mathcal{D}_1(a)(\rho_D)$ é

  $$\mathcal{D}_1(a)(\rho_D)[1] = \{(a, \emptyset)\} = \{(a, \mathcal{D}_1(E)(\rho_D)[0])\}$$

  e

  $$\mathcal{D}_1(a)(\rho_D)[n+2] = \{(a, p_\varepsilon)\} = \{(a, \mathcal{D}_1(E)(\rho_D)[n+1])\}$$

  como pretendíamos.

- Caso $s = x$:

  Atendendo à definição de $\mathcal{T}_{syn}$,

  $$x \xrightarrow{a}_D r <=> D(x) \xrightarrow{a}_D r.$$

  Por outro lado, a única possibilidade para $\mathcal{D}_1(x)(\rho_D)$ é

  $$\mathcal{D}_1(x)(\rho_D)[n+1] \overset{def.\mathcal{D}_1}{=} \rho_D(x)[n+1]$$

  que, como ρ_D é o ponto fixo de Ψ_D, é igual a

  $$\mathcal{D}_1(D(x))(\rho_D)[n+1]. \quad (1)$$

  Por hipótese de indução em peso sabemos que

  $$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(D(x))(\rho_D)[n+1] \text{ se e só se } D(x) \xrightarrow{a}_D r$$

e logo, por (1),

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(x)(\rho_D)[n+1] \text{ se e só se } D(x) \xrightarrow{a}_D r$$

donde, atendendo á definição de $\mathcal{T}_{syn}$, se obtém

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(x)(\rho_D)[n+1] \text{ se e só se } x \xrightarrow{a}_D r$$

- caso $s = s_1; s_2$

  Da definição de $\mathcal{T}_{syn}$ temos

  $$s_1; s_2 \xrightarrow{a}_D r_1; s_2 \text{ se e só se } s_1 \xrightarrow{a}_D r_1$$

  que, por hipótese de indução em peso, é equivalente a

  $$(a, \mathcal{D}_1(r_1)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \quad (1)$$

  Ora

  $$\mathcal{D}_1(s_1; s_2)(\rho_D)[n+1] = (\mathcal{D}_1(s_1)(\rho_D))[n+1] \odot_{n+1} (\mathcal{D}_1(s_2)(\rho_D))[n+1]$$

  logo, atendendo a (1) e à definição de $\odot_{n+1}$, vem

  $$(a, \mathcal{D}_1(r_1)(\rho_D)[n] \odot_n \mathcal{D}_1(s_2)(\rho_D)[n]) \in \mathcal{D}_1(s_1; s_2)(\rho_D)[n+1]$$

  o que, atendendo à definição de $\mathcal{D}_1$, é equivalente a

  $$(a, \mathcal{D}_1(r_1; s_2)(\rho_D)[n]) \in \mathcal{D}_1(s_1; s_2)(\rho_D)[n+1]$$

  e demonstra o resultado.

- caso $s = s_1 + s_2$

  Da definição de $\mathcal{T}_{syn}$, sabemos que

  $$s_1 + s_2 \xrightarrow{a}_D r \text{ se e só se } s_1 \xrightarrow{a}_D r \text{ ou } s_2 \xrightarrow{a}_D r$$

que por hipótese de indução em peso é equivalente a

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1]$$

ou ainda

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s_2)(\rho_D)[n+1].$$

Ora

$$\begin{aligned}\mathcal{D}_1(s_1 + s_2)(\rho_D)[n+1] &= \mathcal{D}_1(s_1)(\rho_D)[n+1] \oplus_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1] \\ &= \mathcal{D}_1(s_1)(\rho_D)[n+1] \cup \mathcal{D}_1(s_2)(\rho_D)[n+1]\end{aligned}$$

logo

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s_1 + s_2)(\rho_D)[n+1].$$

Note-se que $\mathcal{D}_1(r)(\rho_D) = \overrightarrow{p_\varepsilon}$ se e só se $r = E$.

- caso $s = s_1 \parallel s_2$

  Da definição de $\mathcal{T}_{syn}$, sabemos que

  $$\begin{aligned} s_1 \parallel s_2 \xrightarrow{a}_D r \text{ se e só se } \quad & s_1 \xrightarrow{a}_D r_1 \\ & \text{ou } s_2 \xrightarrow{a}_D r_2 \\ & \text{ou } s_1 \xrightarrow{c}_D r_1 \text{ e } s_2 \xrightarrow{\overline{c}}_D r_2 \end{aligned}$$

  donde por hipótese de indução em peso

  $$\begin{aligned} & (a, \mathcal{D}_1(r_1)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \text{ e } r = r_1 \parallel s_2 \quad (1) \\ & \text{ou } (a, \mathcal{D}_1(r_2)(\rho_D)[n]) \in \mathcal{D}_1(s_2)(\rho_D)[n+1] \text{ e } r = s_1 \parallel r_2 \quad (2) \\ & \text{ou } (c, \mathcal{D}_1(r_1)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \\ & \qquad \text{e } (\overline{c}, \mathcal{D}_1(r_2)(\rho_D)[n]) \in \mathcal{D}_1(s_2)(\rho_D)[n+1] \\ & \qquad \text{e } r = r_1 \parallel r_2. \qquad (3) \end{aligned}$$

  Sabemos que

  $$\begin{aligned}\mathcal{D}_1(s_1 \parallel s_2)(\rho_D)[n+1] = \; & \mathcal{D}_1(s_1)(\rho_D)[n+1] \circledparallel_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1] \\ = \; & \mathcal{D}_1(s_1)(\rho_D)[n+1] ||\lfloor_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1] \\ & \cup \mathcal{D}_1(s_2)(\rho_D)[n+1] ||\lfloor_{n+1} \mathcal{D}_1(s_1)(\rho_D)[n+1] \\ & \cup \mathcal{D}_1(s_1)(\rho_D)[n+1] \mid_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1].\end{aligned}$$

Para o caso (1) vem,

$$(a, \mathcal{D}_1(r_1)(\rho_D)[n] \textcircled{\scriptsize ||}_n \mathcal{D}_1(s_2)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] ||\lfloor_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1]$$

$$\text{e } r = r_1 \parallel s_2$$

logo

$$(a, \mathcal{D}_1(r_1 \parallel s_2)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] ||\lfloor_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1] \text{ e } r = r_1 \parallel s_2$$

isto é

$$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] ||\lfloor_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1].$$

Para o caso (2) a demonstração faz-se de modo análogo.

Para o caso (3) vem,

$$(\tau, \mathcal{D}_1(r_1)(\rho_D)[n] \textcircled{\scriptsize ||}_n \mathcal{D}_1(r_2)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \mid_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1]$$

$$\text{e } r = r_1 \parallel r_2$$

logo

$$(\tau, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \mid_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1]$$

como pretendíamos.

- Caso $s = s_1 \setminus c$

  Da definição de $\mathcal{T}_{syn}$, sabemos que

  $$s_1 \setminus c \xrightarrow{a}_D r \text{ se e só se } s_1 \xrightarrow{a}_D r \text{ e } a \neq c, \overline{c}$$

  logo, por hipótese de indução em peso

  $$(a, \mathcal{D}_1(r)(\rho_D)[n]) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \text{ e } a \neq c, \overline{c}$$

  e pela definição de $\textcircled{\scriptsize \setminus}_n$

  $$(a, \mathcal{D}_1(r)(\rho_D)[n] \textcircled{\scriptsize \setminus}_n c) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \textcircled{\scriptsize \setminus}_{n+1} c.$$

Como

$$\mathcal{D}_1(s_1 \setminus c)(\rho_D)[n+1] = \mathcal{D}_1(s_1)(\rho_D)[n+1]\oslash_{n+1} c$$

tem-se

$$(a, \mathcal{D}_1(r)(\rho_D)[n]\oslash_n c) \in \mathcal{D}_1(s_1 \setminus c)(\rho_D)[n+1]$$

e logo

$$(a, \mathcal{D}_1(r \setminus c)(\rho_D)[n]) \in \mathcal{D}_1(s_1 \setminus c)(\rho_D)[n+1]$$

o que termina a demonstração. □

Teorema 3.2.16 $\mathcal{O} = \text{abs} \circ \mathcal{D}$.

Demonstração Temos de mostrar que para todo $r \in Res$ tem-se $\mathcal{O}_0(r) = abs \circ \mathcal{D}_0(r)$. A demonstração será feita por indução em peso.

- Caso $r = E$:

$$\begin{aligned}\mathcal{O}_0(E) &= \vec{p_\varepsilon} \\ &= abs \circ \vec{p_\varepsilon} \\ &= abs \circ \mathcal{D}_1(E)(\rho_D) \\ &= abs \circ \mathcal{D}_0(E).\end{aligned}$$

- Caso $r = a$:

  Temos duas situações a considerar. Se $a \in IAct$ vem

$$\begin{aligned}\mathcal{O}_0(a) &= (\mathcal{O}_0(a)[n])_{n\geq 0} \\ &= \vec{a} \\ &= abs \circ \vec{a} \\ &= abs \circ \mathcal{D}_1(a)(\rho_D) \\ &= abs \circ \mathcal{D}_0(a).\end{aligned}$$

  Se $a \notin IAct$ temos

$$\begin{aligned}\mathcal{O}_0(a) &= \emptyset \\ &= abs \circ \vec{a} \\ &= abs \circ \mathcal{D}_1(a)(\rho_D) \\ &= abs \circ \mathcal{D}_0(a).\end{aligned}$$

- Caso $r = x$:

$$\begin{aligned}
\mathcal{O}_0(x) &= (\mathcal{O}_0(x)[n])_{n\geq 0} \\
&= \text{ (por definição de } \mathcal{T}_{syn}) \\
&\quad (\mathcal{O}_0(D(x))[n])_{n\geq 0} \\
&= \mathcal{O}_0(D(x)) \\
&= \text{ (por hipótese de indução em } \mathsf{peso}) \\
&\quad abs \circ \mathcal{D}_0(D(x)) \\
&= \text{ (por definição de } \mathcal{D}_0) \\
&\quad abs \circ \mathcal{D}_1(D(x))(\rho_D) \\
&= (\text{porque } \rho_D \text{ é ponto fixo de } \Psi_D) \\
&\quad abs \circ \rho_D(x) \\
&= \text{ (por definição de } \mathcal{D}_1) \\
&\quad abs \circ \mathcal{D}_1(x)(\rho_D) \\
&= \text{ (por definição de } \mathcal{D}_0) \\
&\quad abs \circ \mathcal{D}_0(x).
\end{aligned}$$

- Caso $r = s_1; s_2$:

  $\mathcal{O}_0(s_1; s_2) = (\mathcal{O}_0(s_1; s_2)[n])_{n\geq 0}$

  e $abs \circ \mathcal{D}_0(s_1; s_2) = ((abs \circ \mathcal{D}_0(s_1 + s_2))[n])_{n\geq 0}$.

  Vamos mostrar por indução em n que

$$(\mathcal{O}_0(s_1; s_2)[n]) = (abs \circ \mathcal{D}_0(s_1; s_2))[n] \text{ para todo o } n \geq 0,$$

  isto é,

$$(\mathcal{O}_0(s_1; s_2)[n]) = (abs_n \circ \mathcal{D}_0(s_1; s_2)[n]) \text{ para todo o } n \geq 0.$$

$$\begin{aligned}
\mathcal{O}_0(s_1; s_2)[0] &= \emptyset \\
&= abs_0 \circ \mathcal{D}_0(s_1; s_2)[0]
\end{aligned}$$

  Consideremos agora o caso $n + 1$. Por um lado temos,

$$
\begin{array}{rcl}
\mathcal{O}_0(s_1;s_2)[n+1] & = & \{(b,\mathcal{O}_0(r)[n]) : s_1;s_2 \xrightarrow{b}_D r\} \\
& = & \text{(por definição de } \mathcal{T}_{syn}) \\
& & \{(b,\mathcal{O}_0(r_1;s_2)[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& = & \text{(por hipótese de indução em } n) \\
& & \{(b,(abs\circ\mathcal{D}_0(r_1;s_2))[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& = & \text{(por definição de } \mathcal{D}_0\&abs) \\
& & \{(b,abs_n\circ(\mathcal{D}_1(r_1;s_2)(\rho_D))[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& = & \text{(por definição de } \mathcal{D}_1) \\
& & \{(b,abs_n\circ(\mathcal{D}_1(r_1)(\rho_D)\dot{\oplus}\mathcal{D}_1(s_2)(\rho_D))[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& = & \text{(por definição de } \dot{\oplus}) \\
& & \{(b,abs_n\circ(\mathcal{D}_1(r_1)(\rho_D)[n]\dot{\oplus}_n\mathcal{D}_1(s_2)(\rho_D))[n] : \\
& & \qquad\qquad s_1 \xrightarrow{b}_D r_1\}.
\end{array}
$$

Por outro lado,

$$
\begin{array}{rcl}
abs_{n+1}\circ\mathcal{D}_0(s_1;s_2)[n+1] & = & abs_{n+1}\circ\mathcal{D}_1(s_1;s_2)(\rho_D)[n+1] \\
& = & \text{(por definição de } \mathcal{D}_1) \\
& & abs_{n+1}\circ((\mathcal{D}_1(s_1)(\rho_D)\dot{\oplus}\mathcal{D}_1(s_2)(\rho_D))[n+1] \\
& = & \text{(por definição de } \dot{\oplus}) \\
& & abs_{n+1}\circ\{(a,x\dot{\oplus}_n\beta_n(\mathcal{D}_1(s_2)(\rho_D))[n+1])) : \\
& & \qquad\qquad (a,x)\in\mathcal{D}_1(s_1)(\rho_D))[n+1]\} \\
& = & \text{(por definição de } \beta_n \text{ \& lema 3.2.15)} \\
& & abs_{n+1}\circ\{(a,x\dot{\oplus}_n\mathcal{D}_1(s_2)(\rho_D))[n]) : \\
& & \qquad\qquad s_1 \xrightarrow{a}_D r_1 \text{ e } x=\mathcal{D}_1(r_1)(\rho_D))[n]\} \\
& = & abs_{n+1}\circ\{(a,\mathcal{D}_1(r_1)(\rho_D))[n]\dot{\oplus}_n\mathcal{D}_1(s_2)(\rho_D))[n]) : \\
& & \qquad\qquad s_1 \xrightarrow{a}_D r_1\} \\
& = & \text{(por definição de } abs) \\
& & \{(b,abs_n\circ(\mathcal{D}_1(r_1)(\rho_D))[n]\dot{\oplus}_n\mathcal{D}_1(s_2)(\rho_D))[n])) : \\
& & \qquad\qquad s_1 \xrightarrow{b}_D r_1 \text{ e } b\in IAct\},
\end{array}
$$

como pretendiamos.

- Caso $r = s_1 + s_2$: Temos

$\mathcal{O}_0(s_1 + s_2) = (\mathcal{O}_0(s_1 + s_2)[n])_{n\geq 0}$
e $abs \circ \mathcal{D}_0(s_1 + s_2) = ((abs \circ \mathcal{D}_0(s_1 + s_2))[n])_{n\geq 0}$.
Vamos mostrar por indução em n que
$(\mathcal{O}_0(s_1 + s_2)[n]) = (abs \circ \mathcal{D}_0(s_1 + s_2))[n]$ para todo o $n \geq 0$,
isto é,
$(\mathcal{O}_0(s_1 + s_2)[n]) = (abs_n \circ \mathcal{D}_0(s_1 + s_2)[n])$ para todo o $n \geq 0$.

$$\begin{aligned}\mathcal{O}_0(s_1 + s_2)[0] &= \emptyset \\ &= abs_0 \circ (\mathcal{D}_0(s_1 + s_2)[0]).\end{aligned}$$

$$\begin{aligned}\mathcal{O}_0(s_1 + s_2)[n+1] &= \{(b, \mathcal{O}_0(r)[n]) : s_1 + s_2 \xrightarrow{b}_D r\} \\ &= \text{(por definição de } \mathcal{T}_{syn}) \\ &\quad \{(b, \mathcal{O}_0(r_1)[n]) : s_1 \xrightarrow{b}_D r_1\} \\ &\quad \cup \\ &\quad \{(b, \mathcal{O}_0(r_2)[n]) : s_2 \xrightarrow{b}_D r_2\} \\ &= \text{(por definição de } \mathcal{O}_0) \\ &\quad \mathcal{O}_0(s_1)[n+1] \cup \mathcal{O}_0(s_2)[n+1] \\ &= \text{(po hipótese de indução em peso)} \\ &\quad (abs_{n+1} \circ \mathcal{D}_0(s_1))[n+1] \cup (abs_{n+1} \circ \mathcal{D}_0(s_2))[n+1] \\ &= \text{(por definição de } abs) \\ &\quad abs_{n+1} \circ (\mathcal{D}_0(s_1)[n+1] \cup \mathcal{D}_0(s_2))[n+1]) \\ &= \text{(por definição de } \mathcal{D}_0) \\ &\quad abs_{n+1} \circ ((\mathcal{D}_1(s_1)(\rho_D))[n+1] \cup (\mathcal{D}_1(s_2)(\rho_D))[n+1]) \\ &= \text{(por definição de } \oplus) \\ &\quad abs_{n+1} \circ ((\mathcal{D}_1(s_1)(\rho_D) \oplus \mathcal{D}_1(s_2)(\rho_D))[n+1] \\ &= \text{(por definição de } \mathcal{D}_1) \\ &\quad abs_{n+1} \circ (\mathcal{D}_1(s_1 + s_2)\rho_D)[n+1] \\ &= (abs \circ \mathcal{D}_0(s_1 + s_2))[n+1].\end{aligned}$$

- Caso $r = s_1 \parallel s_2$:
  Temos $\mathcal{O}_0(s_1 \parallel s_2) = (\mathcal{O}_0(s_1 \parallel s_2)[n])_{n\geq 0}$
  e $abs \circ \mathcal{D}_0(s_1 \parallel s_2) = ((abs \circ \mathcal{D}_0(s_1 \parallel s_2))[n])_{n\geq 0}$.

Vamos mostrar por indução em n que
$(\mathcal{O}_0(s_1 \parallel s_2)[n]) = (abs \circ \mathcal{D}_0(s_1 \parallel s_2))[n]$ para todo o $n \geq 0$,
isto é, $(\mathcal{O}_0(s_1 \parallel s_2)[n]) = (abs_n \circ \mathcal{D}_0(s_1 \parallel s_2)[n])$ para todo o $n \geq 0$.

$$
\begin{aligned}
\mathcal{O}_0(s_1 \parallel s_2)[0] = \;& \emptyset \\
=\; & abs_0 \circ (\mathcal{D}_0(s_1 \parallel s_2)[0]).
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{O}_0(s_1 \parallel s_2)[n+1] = \;& \{(b, \mathcal{O}_0(r)[n]) : s_1 \parallel s_2 \xrightarrow{b}_D r\} \\
=\; & \text{(por definição de } \mathcal{T}_{syn}) \\
& \{(b, \mathcal{O}_0(r_1 \parallel s_2)[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& \cup \\
& \{(b, \mathcal{O}_0(s_1 \parallel r_2)[n]) : s_2 \xrightarrow{b}_D r_2\} \\
& \cup \\
& \{(\tau, \mathcal{O}_0(r_1 \parallel r_2)[n]) : s_1 \xrightarrow{c}_D r_1 \text{ e } s_2 \xrightarrow{\overline{c}}_D r_2\} \\
=\; & \text{(por hipótese de indução em } n) \\
& \{(b, (abs \circ \mathcal{D}_0(r_1 \parallel s_2))[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& \cup \\
& \{(b, (abs \circ \mathcal{D}_0(s_1 \parallel r_2))[n]) : s_2 \xrightarrow{b}_D r_2\} \\
& \cup \\
& \{(\tau, (abs \circ \mathcal{D}_0(r_1 \parallel r_2))[n]) : s_1 \xrightarrow{c}_D r_1 \text{ e } s_2 \xrightarrow{\overline{c}}_D r_2\} \\
=\; & \text{(por definição de } \mathcal{D}_0 \& abs) \\
& \{(b, (abs_n \circ (\mathcal{D}_1(r_1)(\rho_D)[n] \textcircled{\scriptsize ||}_n \mathcal{D}_1(s_2)(\rho_D)[n]) : s_1 \xrightarrow{b}_D r_1\} \\
& \cup \\
& \{(b, (abs_n \circ (\mathcal{D}_1(s_1)(\rho_D)[n] \textcircled{\scriptsize ||}_n \mathcal{D}_1(r_2)(\rho_D)[n]) : s_2 \xrightarrow{b}_D r_2\} \\
& \cup \\
& \{(\tau, (abs_n \circ (\mathcal{D}_1(r_1)(\rho_D)[n] \textcircled{\scriptsize ||}_n \mathcal{D}_1(r_2)(\rho_D)[n]) : \\
& \qquad\qquad s_1 \xrightarrow{c}_D r_1 \text{ e } s_2 \xrightarrow{\overline{c}}_D r_2\}.
\end{aligned}
$$

Por outro lado tem-se,

$$
\begin{aligned}
abs_{n+1} \circ \mathcal{D}_0(s_1 \parallel s_2)[n+1] = \;& abs_{n+1} \circ \mathcal{D}_1(s_1 \parallel s_2)(\rho_D)[n+1] \\
= \;& \text{(por definição de } \mathcal{D}_1\text{)} \\
& abs_{n+1} \circ (\mathcal{D}_1(s_1)(\rho_D) \textcircled{\scriptsize ||} \mathcal{D}_1(s_2)(\rho_D))[n+1] \\
= \;& \text{(por definição de } \textcircled{\scriptsize ||}\text{)} \\
& abs_{n+1} \circ (\mathcal{D}_1(s_1)(\rho_D)[n+1] \textcircled{\scriptsize ||}_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1]) \\
= \;& \text{(por definição de } \textcircled{\scriptsize ||}_{n+1} \& abs\text{)} \\
& abs_{n+1} \circ (\mathcal{D}_1(s_1)(\rho_D)[n+1] \; ||\lfloor_{n+1} \mathcal{D}_1(s_2)(\rho_D)[n+1]) \\
& \cup \\
& abs_{n+1} \circ (\mathcal{D}_1(s_2)(\rho_D)[n+1] \; ||\lfloor_{n+1} \mathcal{D}_1(s_1)(\rho_D)[n+1]) \\
& \cup \\
& abs_{n+1} \circ (\mathcal{D}_1(s_2)(\rho_D)[n+1] \; |_{n+1} \mathcal{D}_1(s_1)(\rho_D)[n+1]) \\
= \;& \text{(por definição de } ||\lfloor_{n+1} \& \; |_{n+1}\text{)} \\
& abs_{n+1} \circ (\{(a, x \textcircled{\scriptsize ||}_n \beta_n(\mathcal{D}_1(s_2)(\rho_D)[n+1]) : \\
& \qquad\qquad (a, x) \in \mathcal{D}_1(s_1)(\rho_D)[n+1]\} \\
& \cup \\
& abs_{n+1} \circ (\{(a, x \textcircled{\scriptsize ||}_n \beta_n(\mathcal{D}_1(s_1)(\rho_D)[n+1]) : \\
& \qquad\qquad (a, x) \in \mathcal{D}_1(s_2)(\rho_D)[n+1]\} \\
& \cup \\
& abs_{n+1} \circ (\{(\tau, x \textcircled{\scriptsize ||}_n y) : (c, x) \in \mathcal{D}_1(s_1)(\rho_D)[n+1] \\
& \qquad\qquad \text{e } (\overline{c}, y) \in \mathcal{D}_1(s_2)(\rho_D)[n+1]\} \\
= \;& \text{(pelo lema 3.2.15)} \\
& abs_{n+1} \circ (\{(a, x \textcircled{\scriptsize ||}_n \beta_n(\mathcal{D}_1(s_2)(\rho_D)[n+1]) : \\
& \qquad\qquad s_1 \xrightarrow{a}_D r_1, x = \mathcal{D}_1(r_1)(\rho_D)[n]\} \\
& \cup \\
& abs_{n+1} \circ (\{(a, x \textcircled{\scriptsize ||}_n \beta_n(\mathcal{D}_1(s_1)(\rho_D)[n+1]) : \\
& \qquad\qquad s_2 \xrightarrow{a}_D r_2, x = \mathcal{D}_1(r_2)(\rho_D)[n]\} \\
& \cup \\
& abs_{n+1} \circ (\{(\tau, x \textcircled{\scriptsize ||}_n y) : s_1 \xrightarrow{c}_D r_1, s_2 \xrightarrow{\overline{c}}_D r_2, \\
& \qquad\qquad x = \mathcal{D}_1(r_1)(\rho_D)[n]\}, y = \mathcal{D}_1(r_2)(\rho_D)[n]\}
\end{aligned}
$$

$$
\begin{aligned}
= \quad & \text{(por definição de } \beta_n\text{)} \\
& abs_{n+1} \circ (\{(a, \mathcal{D}_1(r_1)(\rho_D)[n] \circledparallel_n \mathcal{D}_1(s_2)(\rho_D)[n]) : \\
& \qquad\qquad s_1 \xrightarrow{a}_D r_1\} \\
& \cup \\
& abs_{n+1} \circ (\{(a, \mathcal{D}_1(r_2)(\rho_D)[n] \circledparallel_n \mathcal{D}_1(s_1)(\rho_D)[n]) : \\
& \qquad\qquad s_2 \xrightarrow{a}_D r_2\} \\
& \cup \\
& abs_{n+1} \circ (\{(\tau, \mathcal{D}_1(r_1)(\rho_D)[n] \circledparallel_n \mathcal{D}_1(r_2)(\rho_D)[n]) : \\
& \qquad s_1 \xrightarrow{c}_D r_1, s_2 \xrightarrow{\bar{c}}_D r_2\}.
\end{aligned}
$$

O que aplicando a definição de abs dá

$$
\begin{aligned}
abs_{n+1} \circ \mathcal{D}_0(s_1 \parallel s_2)[n+1] = \quad & \{(b, \mathcal{D}_1(r_1)(\rho_D)[n] \circledparallel_n \mathcal{D}_1(s_2)(\rho_D)[n]) : \\
& \qquad\qquad s_1 \xrightarrow{b}_D r_1, b \in IAct\} \\
& \cup \\
& \{(b, \mathcal{D}_1(r_2)(\rho_D)[n] \circledparallel_n \mathcal{D}_1(s_1)(\rho_D)[n]) : \\
& \qquad\qquad s_2 \xrightarrow{b}_D r_2, b \in IAct\} \\
& \cup \\
& \{(\tau, \mathcal{D}_1(r_1)(\rho_D)[n] \circledparallel_n \mathcal{D}_1(r_2)(\rho_D)[n]) : \\
& \qquad\qquad s_1 \xrightarrow{c}_D r_1, s_2 \xrightarrow{\bar{c}}_D r_2\}
\end{aligned}
$$

como pretendíamos.

- Caso $r = s \setminus c$:

  $\mathcal{O}_0(s \setminus c) = (\mathcal{O}_0(s \setminus c)[n])_{n \geq 0}$

  e $abs \circ \mathcal{D}_0(s \setminus c) = ((abs \circ \mathcal{D}_0(s \setminus c))[n])_{n \geq 0}$.

  Vamos mostrar por indução em n que

  $$(\mathcal{O}_0(s \setminus c)[n]) = (abs_n \circ \mathcal{D}_0(s \setminus c)[n]) \text{ para todo o } n \geq 0.$$

  $$
  \begin{aligned}
  \mathcal{O}_0(s \setminus c)[0] = \quad & \emptyset \\
  = \quad & abs_0 \circ (\mathcal{D}_0(s \setminus c)[0]).
  \end{aligned}
  $$

$$\begin{aligned}
\mathcal{O}_0(s \setminus c)[n+1] = &\ \{(b, \mathcal{O}_0(r_1)[n]) : s \setminus c \xrightarrow{b}_D r_1\} \\
= &\ \text{(por definição de } \mathcal{T}_{syn},\ b \in IAct) \\
&\ \{(b, \mathcal{O}_0(r \setminus c)[n]) : s \xrightarrow{b}_D r\} \\
= &\ \text{(por hipótese de indução em } n) \\
&\ \{(b, abs_n \circ \mathcal{D}_0(r \setminus c)[n]) : s \xrightarrow{b}_D r\}.
\end{aligned}$$

Por outro lado temos

$$\begin{aligned}
abs_{n+1} \circ \mathcal{D}_0(s \setminus c)[n+1] = &\ abs_{n+1} \circ \mathcal{D}_1(s \setminus c)(\rho_D)[n+1] \\
= &\ \text{(por definição de } \mathcal{D}_1) \\
&\ abs_{n+1} \circ (\mathcal{D}_1(s)(\rho_D) \textcircled{\setminus} c)[n+1] \\
= &\ \text{(por definição de } \textcircled{\setminus}) \\
&\ abs_{n+1} \circ \{(a, x \textcircled{\setminus}_n c) : (a, x) \in \mathcal{D}_1(s)(\rho_D)[n+1], \\
&\qquad\qquad a \neq c, \overline{c}\} \\
= &\ \text{(pelo lema 3.2.15)} \\
&\ abs_{n+1} \circ \{(a, \mathcal{D}_1(r)(\rho_D)[n] \textcircled{\setminus}_n c) : s \xrightarrow{a}_D r, a \neq c, \overline{c}\} \\
= &\ \text{(por definição de } \mathcal{D}_1) \\
&\ abs_{n+1} \circ \{(a, \mathcal{D}_1(r \setminus c)(\rho_D)[n]) : s \xrightarrow{a}_D r, a \neq c, \overline{c}\} \\
= &\ \text{(por definição de } \mathcal{D}_0) \\
&\ abs_{n+1} \circ \{(a, \mathcal{D}_0(r \setminus c)[n]) : s \xrightarrow{a}_D r, a \neq c, \overline{c}\} \\
= &\ \text{(por definição de } abs) \\
&\ \{(b, abs_n \circ \mathcal{D}_0(r \setminus c)[n]) : s \xrightarrow{b}_D r\}
\end{aligned}$$

como pretendíamos.

Com isto finalizamos a demonstração. □

3.3 Semântica utilizando coálgebras

Nesta secção vamos utilizar autómatos sob a forma de coálgebras e definir operações análogas às existentes na linguagem $\mathcal{L}_{syn}$ como operações entre autómatos (coálgebras para um functor apropriado). Com base nessas operações entre autómatos e na existência de uma coálgebra final vamos definir a semântica denotacional da linguagem. A semântica operacional é também definida à custa do morfismo na coálgebra final, mas para um sistema baseado nas transições de $\mathcal{T}_{syn}$.

3.3.1 Domínio semântico

Uma das vantagens da abordagem coalgébrica é que a categoria de base, na qual os funtores estão definidos, é a categoria dos conjuntos e não a dos Cfe's. Em particular, os domínios que vamos utilizar são pontos fixos do functor

$$H(Q) = \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times Q)$$

onde A é um conjunto de acções que será $IAct$ no caso da semântica operacional e Act no caso da semântica denotacional.

Dada a sucessão de conjuntos $(\mathcal{U}_n)_{n\geq o}$, definida por

$$\begin{aligned}&\mathcal{U}_0 = \{\emptyset\}\\ &\mathcal{U}_{n+1} = H(\mathcal{U}_n) = \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times \mathcal{U}_n)\end{aligned}$$

e as funções

$$\begin{aligned}&\beta_0 : \mathcal{U}_1 \to \mathcal{U}_0 \text{ a única função possível e}\\ &\beta_{n+1} = H(\beta_n) : \mathcal{U}_{n+2} \to \mathcal{U}_{n+1}\end{aligned}$$

onde

$$\begin{aligned}&\beta_{n+1}(p_\varepsilon) = p_\varepsilon\\ &\beta_{n+1}(X) = \{(a, \beta_n(u)) : (a, u) \in X\}\end{aligned}$$

define-se o seu limite

$$\mathcal{U} = \{(u_n)_{n\geq 0} : \forall_n\ u_n \in \mathcal{U}_n \text{ e } u_n = \beta_n(u_{n+1})\}$$

Relativamente aos elementos de $\mathcal{U}$ iremos utilizar as notações $u = (u_n)_{n\geq 0}$ e $u[n] = u_n$.

Consideremos em Set a coálgebra $(\mathcal{U}, \xi)$ com

$$\xi : \mathcal{U} \to \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times \mathcal{U})$$

onde

$$\xi(\overrightarrow{p_\varepsilon}) = p_\varepsilon$$
$$\xi(u) = \{(a, v) : \forall n\ (a, v[n]) \in u[n+1]\}.$$

Usamos $u \downarrow$ para denotar $\xi(u) = p_\varepsilon$, isto é $u \downarrow$ se e só se $u = \overrightarrow{p_\varepsilon} = (\overrightarrow{p_\varepsilon}[n])_{n\geq 0}$ onde $\overrightarrow{p_\varepsilon}[0] = \emptyset$, $\overrightarrow{p_\varepsilon}[n+1] = p_\varepsilon$. Escrevemos $u \xrightarrow{a} v$ se e só se $(a, v) \in \xi(u)$.

Definição 3.3.1 (Finitamente ramificado) Dado $X \subseteq \mathcal{U}$ diz-se que X é *finitamente ramificado* se

$$\forall_{u\in\mathcal{U}},\ u \in X \Longrightarrow \begin{cases} \xi(u) = \overrightarrow{p_\varepsilon} \text{ ou} \\ \xi(u) \subseteq_{fin} A \times X \end{cases}$$

□

Assim, o conjunto vazio, $\emptyset$, é finitamente ramificado e se todo o elemento da família $X_i (i \in I)$ é finitamente ramificado então $\cup_{i\in I} X_i$ é finitamente ramificado.

Definição 3.3.2 (Domínio $\mathbb{T}$) O conjunto $\mathbb{T}$ é o maior subconjunto de $\mathcal{U}$ finitamente ramificado. □

A coálgebra final que nos interessa é definida com base em ξ mas apenas sobre os elementos finitamente ramificados.

Definição 3.3.3 $\iota : \mathbb{T} \to \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times \mathbb{T})$ é definida por

$$\iota(t) = \xi(t).$$

□

Teorema 3.3.4 *O par* $(\mathbb{T}, \iota)$ *é uma coálgebra final para o functor* $H(Q) = \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times Q)$.

Demonstração Temos de provar que para todo o par (Q, ϕ) com $\phi : Q \to \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times Q)$ existe um único morfismo $f : (Q, \phi) \to (\mathbb{T}, \iota)$. Vamos estruturar a demonstração em quatro pontos: definimos a função f, mostramos que está bem definida, provamos que é um morfismo de coálgebras e por fim que é único.

1. Definimos $f : Q \to \mathbb{T}$ por

$$f(q) = \begin{cases} \overrightarrow{p_\varepsilon} & \text{se} \quad q \downarrow_\phi \\ \overrightarrow{q} = (\overrightarrow{q}\ [n])_{n \geq 0} & \text{se} \quad \phi(q) \subseteq_{fin} A \times Q \end{cases}$$

   onde $\overrightarrow{q}\ [0] = \emptyset$, $\overrightarrow{q}\ [n+1] = \{(a, f(p)[n]) : (a, p) \in \phi(q)\}$.

2. Para mostrar que f está bem definida temos de verificar que

   (a) $\overrightarrow{q} \in \mathcal{U}$. Para isto temos de ter:

      i. $\overrightarrow{q}\ [n] \in \mathcal{U}_n$, para todo o n.
      $\overrightarrow{q}\ [0] = \emptyset\ \in \mathcal{U}_0$
      $\overrightarrow{q}\ [n+1] \subseteq_{fin} A \times \mathcal{U}_n$, por hipótese de indução, logo
      $\overrightarrow{q}\ [n+1] \in \mathcal{U}_{n+1}$.

      ii. $\beta_n(\overrightarrow{q}\ [n+1]) = \overrightarrow{q}\ [n]$, para todo o n

$$\begin{aligned} \beta_0(\overrightarrow{q}\ [1]) &= \emptyset \\ &= \overrightarrow{q}\ [0] \\ \beta_{n+1}(\overrightarrow{q}\ [n+2]) &= \beta_{n+1}(\{(a, f(p)[n+1]) : (a, p) \in \phi(q)\}) \\ &= \{(a, \beta_n(f(p)[n+1])) : (a, p) \in \phi(q)\} \\ &= \text{(por hipótese de indução)} \\ &\quad \{(a, f(p)[n]) : (a, p) \in \phi(q)\} \\ &= \overrightarrow{q}\ [n+1] \end{aligned}$$

   (b) $f(Q) \subseteq \mathbb{T}$, isto é, $f(Q) = \{f(q) : q \in Q\}$ é finitamente ramificado.

É preciso provar que $\forall_q,\ \xi(f(q)) = p_\varepsilon$ ou $\xi(f(q)) \subseteq_{fin} A \times f(Q)$. Se $\xi(f(q)) = p_\varepsilon$ não há nada a demonstrar.

Suponhamos que $f(q) = \overrightarrow{q} \neq \overrightarrow{p_\varepsilon}$. Primeiro vamos mostrar que se tivermos $\overrightarrow{q} \xrightarrow{a} u$ em $\mathcal{U}$, então existe $p \in Q$ tal que $q \xrightarrow{a}_\phi p$ e $u = f(p)$.

Mas, se $\overrightarrow{q} \xrightarrow{a} u$ isto é equivalente a dizermos que para todo o n tem-se $(a, u[n]) \in \overrightarrow{q}\ [n+1]$. Por outro lado, da definição de $\overrightarrow{q}$ $[n+1]$, sabemos que existe p_n tal que $q \xrightarrow{a}_\phi p_n$ e $u[n] = f(p_n)[n]$. Ora, os elementos $(a, p_1), ..., (a, p_n), ...$ pertencem todos a $\phi(q)$. Como $\phi(q)$ é finito, pelo menos um dos elementos $p_1, ..., p_n, ...$ aparece infinitamente repetido, chamemos p a esse elemento. É fácil ver que qualquer que seja n tem-se $u[n] = f(p)[n]$. (Porque se $u[n+1] = f(p)[n+1]$ então $u[n] = \beta_n(u[n+1]) = \beta_n(f(p)[n+1]) = f(p)[n]$ por definição de $\mathcal{U}$.) Isto implica $\forall_n,\ u \equiv_n f(p)$, logo $u = f(p)$ porque $\mathcal{U}$ é separado. Segue-se que

$$\xi(\overrightarrow{q}) = \{(a, f(p)) : (a, p) \in \phi(q)\},$$

logo é finito porque $\phi(q)$ é finito e está contido em $A \times f(Q)$.

3. Vamos agora mostrar que $f : (Q, \phi) \to (\mathbb{T}, \iota)$ é um morfismo de coálgebras. Para tal, temos de verificar que o diagrama seguinte, onde f^* denota $H(f)$, comuta

$$\begin{array}{ccc} Q & \xrightarrow{\quad f \quad} & \mathbb{T} \\ {\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle\iota} \\ \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times Q) & \xrightarrow{f^*} & \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times \mathbb{T}) \end{array}$$

Para o caso $\phi(q) = p_\varepsilon$ temos $f^*(\phi(q)) = f^*(p_\varepsilon) = p_\varepsilon$ e, por outro lado, $f(q) = \overrightarrow{p_\varepsilon}$ o que implica $\iota(f(q)) = \iota(\overrightarrow{p_\varepsilon}) = \xi(\overrightarrow{p_\varepsilon}) = pe$.

Para o caso $\phi(q) \subseteq A \times Q$ temos $f^*(\phi(q)) = \{(a, f(p)) : (a, p) \in \phi(q)\}$ e, por outro lado, $\iota(f(q)) = \xi(\overrightarrow{q}) = \{(a, u) : \overrightarrow{q} \xrightarrow{a} u\} = \{(a, f(p)) :$

$q \xrightarrow{a}_{\phi} p\} = \{(a, f(p)) : (a, p) \in \phi(q)\}$.

4. Falta-nos mostrar que f é único, isto é, se $g : (Q, \phi) \to (\mathbb{T}, \iota)$ for um morfismo então $g = f$. Consideremos o diagrama

$$\begin{array}{ccc} Q & \xrightarrow{\quad g \quad} & \mathbb{T} \\ {\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle\iota} \\ \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times Q) & \xrightarrow{g^*} & \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times \mathbb{T}) \end{array}$$

Para o caso $\phi(q) = p_\varepsilon$ temos $g^*(\phi(q)) = p_\varepsilon$. Logo $\iota(g(q)) = p_\varepsilon$, donde $g(q) = \overrightarrow{p_\varepsilon} = f(q)$ como pretendíamos.

No caso em que $\phi(q) \subseteq A \times Q$ sabemos que

(i) se $(a, p) \in \phi(q)$ então $\forall_n\ (a, g(p)[n]) \in g(q)[n+1]$;

(ii) se $\forall_v(\forall_n\ (a, v[n]) \in g(q)[n+1])$ então existe p tal que $(a, p) \in \phi(q)$ e $v = g(p)$.

Assim,

$$\begin{aligned} g^*(\phi(q)) &= \{(a, g(p)) : (a, p) \in \phi(q)\} \\ &= \{(a, g(p)) : (a, v[n]) \in g(q)[n+1]\} \\ &= \xi(g(q)) = \iota(g(q)) \end{aligned}$$

Por outro lado se $\phi(q) \subseteq A \times Q$ temos que $f(q) = \overrightarrow{q}$. Vamos mostrar por indução em n que $\forall_n\ \forall_q,\ g(q)[n] = \overrightarrow{q}\ [n]$, o que implica que $g(q) = \overrightarrow{q} = f(p)$, como pretendemos.

Para $n = 0$ temos $\forall_q,\ g(q)[n] = \emptyset = \overrightarrow{q}\ [0]$.

Para $n + 1$ vem

$$\begin{aligned} \forall_q,\ g(q)[n+1] = &\ \{(a, g(p)[n]) : (a, p) \in \phi(q)\} \\ = &\ \{(a, f(p)[n]) : (a, p) \in \phi(q)\} && \text{(por hipótese de indução)} \\ = &\ f(q)[n+1] && \text{(por definição de } f(q)). \end{aligned}$$

Com isto fica completa a demonstração. □

3.3.2 Modelos e Operações

Os modelos a considerar, para a definição da semântica de $\mathcal{L}_{syn}$ são autómatos M definidos por um par (Q, ϕ) em que Q é o conjunto de estados e ϕ é uma função, designada por dinâmica do autómato, que tem tipo

$$\phi : Q \to \{p_\varepsilon\} + \mathcal{P}_{fin}(Act \times Q);$$

isto é, dado um estado q, $\phi(q) = p_\varepsilon$ ou $\phi(q) \subseteq_{fin} Act \times Q$. No primeiro caso estamos perante uma terminação e no segundo caso $\phi(q)$ é o conjunto dos pares (a, q') tais que é possível transitar de q para q' tendo como acção observável a.

Vamos utilizar a seguinte notação,

$$q \xrightarrow{a}_\phi p \Leftrightarrow (a, p) \in \phi(q);$$

$$q \downarrow_\phi \Leftrightarrow \phi(q) = p_\varepsilon.$$

De seguida define-se, sobre os sistemas do tipo $M = (Q, \phi)$, operações correspondentes às existentes entre os processos da linguagem em estudo, $\mathcal{L}_{syn}$. Iremos seguir as notações usuais para os conjuntos e seus elementos: $a \in Act, b \in IAct$ e $c \in Sync$. Vamos utilizar o símbolo $\triangledown$ para denotar um estado final.

Acção

$M_a = (\{*, \triangledown\}, \phi_a)$ onde ϕ_a é definida por

$$(1) \quad * \xrightarrow{a}_{\phi_a} \triangledown$$

$$(2) \quad \triangledown \downarrow_{\phi_a}$$

Composição paralela

Dados $M = (Q, \phi)$ e $N = (P, \psi)$, a sua *composição paralela* é

$$M \parallel N = (Q \times P, \phi \parallel \psi)$$

em que $\phi \parallel \psi$ é definida por

$$(1) \quad (q,p)\downarrow_{\phi\|\psi} \quad \text{se e só se } q\downarrow_{\phi} \text{ e } p\downarrow_{\psi}$$

$$(2) \quad \frac{q \xrightarrow{c}_{\phi} q' \qquad p \xrightarrow{\overline{c}}_{\psi} p'}{(q,p) \xrightarrow{\tau}_{\phi\|\psi} (q',p')}$$

$$(3) \quad \frac{q \xrightarrow{a}_{\phi} q'}{(q,p) \xrightarrow{a}_{\phi\|\psi} (q',p)}$$

$$(4) \quad \frac{p \xrightarrow{a}_{\psi} p'}{(q,p) \xrightarrow{a}_{\phi\|\psi} (q,p')}$$

Restrição

Dado $M = (Q,\phi)$ e $c \in Sync$, M *restringido* a c é

$$M \setminus c = (Q, \phi \setminus c)$$

em que $\phi \setminus c$ é definida por

$$(1) \quad p\downarrow_{\phi\setminus c} \quad \text{se e só se } \; p\downarrow_{\phi}$$

$$(2) \quad \frac{p \xrightarrow{a}_{\phi} q, \quad a \notin \{c,\overline{c}\}}{p \xrightarrow{a}_{\phi\setminus c} q}$$

Soma

Dados $M = (Q,\phi)$ e $N = (P,\psi)$ a sua *soma* é

$$M + N = ((Q + \{\triangledown\}) \times (P + \{\triangledown\}), \phi + \psi)$$

em que $\phi + \psi$ é definida por

(1) $(\nabla, p) \downarrow_{\phi+\psi}$ se e só se $p \downarrow_{\psi}$

(2) $(q, \nabla) \downarrow_{\phi+\psi}$ se e só se $q \downarrow_{\phi}$

(3) $(q, p) \downarrow_{\phi+\psi}$ se e só se $q \downarrow_{\phi}$ e $p \downarrow_{\psi}$

(4) $$\frac{q \xrightarrow{a}_{\phi} q'}{(q,p) \xrightarrow{a}_{\phi+\psi} (q', \nabla)}$$

(5) $$\frac{p \xrightarrow{a}_{\psi} p'}{(q,p) \xrightarrow{a}_{\phi+\psi} (\nabla, p')}$$

Composição sequencial

Dados $M = (Q, \phi)$ e $N = (P, \psi)$ a sua *composição sequencial* é

$$M; N = (Q \times P, \phi; \psi)$$

em que $\phi; \psi$ é definida por

(1) $(q, p) \downarrow_{\phi;\psi}$ se e só se $q \downarrow_{\phi}$ e $p \downarrow_{\psi}$

(2) $$\frac{p \xrightarrow{a}_{\psi} p'}{(q,p) \xrightarrow{a}_{\phi;\psi} (q, p')} \text{ se } q \downarrow_{\phi}$$

(3) $$\frac{q \xrightarrow{a}_{\phi} q'}{(q,p) \xrightarrow{a}_{\phi;\psi} (q', p)}$$

3.3.3 Semântica operacional

As coálgebras que vamos utilizar para a construção da semântica operacional são definidas sobre o functor H introduzido na secção 3.3.1, onde se mostrou também que existe uma coálgebra final para o functor. Dada uma coálgebra-H definida com base no sistema de transições que apresentamos em 3.1 com a

designação $\mathcal{T}_{syn}$, a semântica operacional é obtida através do único morfismo dessa coálgebra na coálgebra final. Para a semântica operacional o conjunto de acções que nos interessa considerar na definição do functor H é $IAct$. O domínio da semântica operacional é o conjunto $\mathbb{T}_{\mathcal{O}}$, solução única da seguinte equação de domínios

$$\mathbb{T}_{\mathcal{O}} \cong \{p_\varepsilon\} + \mathcal{P}_{fin}(IAct \times \mathbb{T}_{\mathcal{O}})$$

como se viu na secção 3.3.1.

Assim, consideremos a coálgebra final $(\mathbb{T}_{\mathcal{O}}, \iota_{\mathcal{O}})$ em Set, com

$$\iota_{\mathcal{O}} : \mathbb{T}_{\mathcal{O}} \to \{p_\varepsilon\} + \mathcal{P}_{fin}(IAct \times \mathbb{T}_{\mathcal{O}}).$$

Associada a $\mathcal{T}_{syn}$ está a coálgebra $(Decl \times Res, \zeta_{syn})$ em que ζ_{syn} é dada por

$$\zeta_{syn} : Decl \times Res \to \{p_\varepsilon\} + \mathcal{P}_{fin}(IAct \times Decl \times Res)$$

$$\begin{aligned} \zeta_{syn}(E) &= p_\varepsilon \\ \zeta_{syn}(s) &= \{(b, r) : s \xrightarrow{b}_D r\} \end{aligned}$$

Como é usual, omitimos a referência a $D \in Decl$ sempre que não haja perigo de confusão. Como os elementos de $\{p_\varepsilon\}$ e $\mathcal{P}_{fin}(IAct \times Decl \times Res)$ são facilmente identificáveis abreviamos $(0, p_\varepsilon)$ e $(1, \{(b_1, r_1)...(b_n, r_n)\})$ para p_ε e $\{(b_1, r_1)...(b_n, r_n)\}$, respectivamente.

Ora, como $(\mathbb{T}_{\mathcal{O}}, \iota_{\mathcal{O}})$ é final, existe um único morfismo

$$\mathcal{O}_0 : (Decl \times Res, \zeta_{syn}) \to (\mathbb{T}_{\mathcal{O}}, \iota_{\mathcal{O}})$$

onde $\mathcal{O}_0 : Decl \times Res \to \mathbb{T}_{\mathcal{O}}$ é dado por

$$\mathcal{O}_0(r) = \begin{cases} \vec{p_\varepsilon} & \text{se } r = E \\ \vec{t} & \text{se } r = s \end{cases}$$

com

$$\begin{aligned} \vec{t}\ [0] &= \emptyset \\ \vec{t}\ [n+1] &= \{(b, \mathcal{O}_0(r)[n]) : s \xrightarrow{b}_D r\}. \end{aligned}$$

Estamos agora em condições de definir a semântica operacional.

Definição 3.3.5 (Semântica operacional) A semântica operacional

$$\mathcal{O}[\![\,.\,]\!] : \mathcal{L}_{syn} \to \mathbb{T}_{\mathcal{O}}$$

é dada por

$$\mathcal{O}[\![\,s\,]\!] = \mathcal{O}_0(s).$$

□

3.3.4 Semântica denotacional

Nesta secção iremos definir a semântica denotacional à custa de operações sobre sistemas da forma $M = (Q, \phi)$. Na definição do domínio semântico temos necessidade de considerar também as acções de sincronização, de forma a preservarmos a composicionalidade. Assim, o domínio da semântica denotacional é a solução da equação de domínios

$$\mathbb{T}_{\mathcal{D}} \cong \{p_\varepsilon\} + \mathcal{P}_{fin}(Act \times \mathbb{T}_{\mathcal{D}}),$$

onde, como sabemos, $Act = IAct \cup Sync$ e consideramos a seguinte coálgebra final em Set, um caso particular de ι, introduzida em 3.3.1, em que o conjunto das acções é Act

$$\iota_{\mathcal{D}} : \mathbb{T}_{\mathcal{D}} \to \{p_\varepsilon\} + \mathcal{P}_{fin}(Act \times \mathbb{T}_{\mathcal{D}}).$$

Para aliviar a notação, nesta secção, iremos utilizar ι em vez de $\iota_{\mathcal{D}}$ e $\mathbb{T}$ em vez de $\mathbb{T}_{\mathcal{D}}$ sempre que não houver perigo de confusão.

Antes de introduzirmos a semântica denotacional é necessário definirmos alguns operadores auxiliares

Definição 3.3.6 (Operadores auxiliares)

(a) Associada à acção a temos $M_a = (\mathbb{T}_a, \phi_a)$ onde $\mathbb{T}_a = \{*, \triangledown\}$. Definimos a função @ como o único morfismo $(\mathbb{T}_a, \phi_a) \to (\mathbb{T}, \iota)$ cuja existência e unicidade está assegurada pelo facto de $(\mathbb{T}, \iota)$ ser final.

Assim, @ é a única função que satisfaz

$$\iota(@(x)) = \begin{cases} p_\varepsilon & \text{se } \phi_a(x) = \triangledown \\ \{(a, @(y)) : (a, y) \in \phi_a(x)\} & \text{se } \phi_a(x) \subseteq \mathcal{P}_{fin}(A \times \mathbb{T}_a). \end{cases}$$

Deste modo temos que

(i) $\iota(@(\triangledown)) = p_\varepsilon$ pelo que, atendendo à definição de ι, vem $@(\triangledown) = \vec{p_\varepsilon}$.

(ii) $\iota(@(*)) = \{(a, @(\triangledown))\} = \{(a, \vec{p_\varepsilon})\}$ pelo que, atendendo à definição de ι, temos $@(*) = \vec{a} = (\vec{a}\ [n])_{n\geq 0}$ com $\vec{a}\ [0] = \emptyset$, $\vec{a}\ [1] = \{(a, \emptyset)\}$ e $\vec{a}\ [n+2] = \{(a, p_\varepsilon)\}$.

(b) Associada à restrição, para todo o $c \in Sync$ definimos a operação $\oslash^c$ como o único morfismo $\oslash^c : (\mathbb{T}, \iota) \setminus c \to (\mathbb{T}, \iota)$.

Assim, $\oslash^c : \mathbb{T} \to \mathbb{T}$ é a única função que satisfaz

$$\iota(\oslash^c(t)) = \begin{cases} p_\varepsilon & \text{se } \iota \setminus c(t) = p_\varepsilon \\ \{(a, \oslash^c(t')) : (a, t') \in \iota \setminus c(t)\} & \text{se } \iota \setminus c(t) \subseteq \mathcal{P}_{fin}(A \times \mathbb{T}) \end{cases}$$

Logo,

$$\oslash^c(t) = \begin{cases} \vec{p_\varepsilon} & \text{se } t = \vec{p_\varepsilon} \\ \vec{u} = (\vec{u}\ [n])_{n\geq 0} & \text{caso contrário} \end{cases}$$

onde $\vec{u}\ [0] = \emptyset$;
$\vec{u}\ [n+1] = \{(a, (\oslash^c(u'))[n]) : (a, u') \in \iota \setminus c(t)\}$

$$= \{(a, (\oslash^c(u'))[n]) : (a, u') \in \iota(t), a \notin \{c, \overline{c}\}\}.$$

(c) Associada à soma $(\mathbb{T}, \iota) + (\mathbb{T}, \iota) = ((\mathbb{T} + \{\triangledown\}) \times (\mathbb{T} + \{\triangledown\}), \iota + \iota)$, definimos a operação $\oplus$ como o único morfismo $\oplus : (\mathbb{T}, \iota) + (\mathbb{T}, \iota) \to (\mathbb{T}, \iota)$.

Assim, $\oplus : (\mathbb{T} + \{\triangledown\}) \times (\mathbb{T} + \{\triangledown\}) \to \mathbb{T}$ satisfaz

$$\iota(u \oplus v) = \begin{cases} p_\varepsilon & \text{se } (\iota + \iota)(u, v) = p_\varepsilon \\ \{(a, u' \oplus v') : (a, (u', v')) \in (\iota + \iota)(u, v)\} & \text{caso contrário.} \end{cases}$$

Deste modo temos que

$$u \oplus v = \begin{cases} \vec{p_\varepsilon} & \text{se } (\iota + \iota)(u, v) = p_\varepsilon \\ \vec{z} = (\vec{z}\ [n])_{n \geq 0} & \text{se } (\iota + \iota)(u, v) \subseteq \mathcal{P}_{fin}(A \times (\mathbb{T} + \{\triangledown\}) \times (\mathbb{T} + \{\triangledown\})) \end{cases}$$

onde $\vec{z}\ [0] = \emptyset$, $\vec{z}\ [n+1] = \{(a, (u' \oplus v')[n]) : (a, (u', v')) \in (\iota + \iota)(u, v)\}$.

(d) Associada a cada uma das operações binárias $_{op} \in \{;, \|\}$ com

$$\iota \,_{op}\, \iota : \mathbb{T} \times \mathbb{T} \to \{p_\varepsilon\} + \mathcal{P}_{fin}(A \times (\mathbb{T} \times \mathbb{T})),$$

definimos a operação $\textcircled{op}$ como o único morfismo
$\textcircled{op} : (\mathbb{T}, \iota) \,_{op}\, (\mathbb{T}, \iota) \to (\mathbb{T}, \iota)$.

Assim, $\textcircled{op} : \mathbb{T} \times \mathbb{T} \to \mathbb{T}$ satisfaz

$$\iota(x \textcircled{op} y) = \begin{cases} p_\varepsilon & \text{se } (\iota \,_{op}\, \iota)(x, y) = p_\varepsilon \\ \{(a, u \textcircled{op} v) : (a, (u, v)) \in (\iota \,_{op}\, \iota)(x, y)\} & \\ & \text{se } (\iota \,_{op}\, \iota)(x, y) \subseteq \mathcal{P}_{fin}(A \times (\mathbb{T} \times \mathbb{T})) \end{cases}$$

Deste modo temos que

$$x \textcircled{op} y = \begin{cases} \vec{p_\varepsilon} & \text{se } (\iota \,_{op}\, \iota)(x, y) = p_\varepsilon \\ \vec{z} = (\vec{z}\ [n])_{n \geq 0} & \text{se } (\iota \,_{op}\, \iota)(x, y) \subseteq \mathcal{P}_{fin}(A \times (\mathbb{T} \times \mathbb{T})) \end{cases}$$

onde $\vec{z}\ [0] = \emptyset$, $\vec{z}\ [n+1] = \{(a, (u \textcircled{op} v)[n]) : (a, (u, v)) \in (\iota \,_{op}\, \iota)(x, y)\}$.

□

Como anteriormente, as declarações $D(\in Decl)$ controlam os ambientes que, neste caso, são definidos por $(\rho \in) Env = PVar \to \mathbb{T}$.

Definição 3.3.7 (Função denotacional auxiliar) O operador

$$\mathcal{D}_1 : Res \to Env \to \mathbb{T}$$

é definido por

$$\begin{aligned}
\mathcal{D}_1(E)(\rho) &= \vec{p_\varepsilon} \\
\mathcal{D}_1(a)(\rho) &= @(*) \ = \vec{a} \\
\mathcal{D}_1(x)(\rho) &= \rho(x) \\
\mathcal{D}_1(s \setminus c)(\rho) &= \diagdown^c \mathcal{D}_1(s)(\rho) \\
\mathcal{D}_1(s_1 \ {}_{op}\ s_2)(\rho) &= \mathcal{D}_1(s_1)(\rho) \ \textcircled{op} \ \mathcal{D}_1(s_1)(\rho)
\end{aligned}$$

onde ${}_{op} \in \{+, ;, \|\}$ e $\vec{a} = (\vec{a})_{n\geq 0}$ com $\vec{a}\ [0] = \emptyset$, $\vec{a}\ [1] = \{(a, \emptyset)\}$, $\vec{a}\ [n+2] = \{(a, p_\varepsilon)\}$ □

A uma declaração $D : PVar \to GStat$ fazemos corresponder um ambiente $\rho_D : PVar \to \mathbb{T}$. Vamos supor aqui que essa correspondência pode ser estabelecida, e relegamos para um apêndice no fim desta secção a construção do ambiente ρ_D a partir da declaração D. Podemos pois considerar que ρ_D é uma função de $PVar$ em $\mathbb{T}$, como se pretendia.

Definição 3.3.8 (Base da semântica denotacional)

$$\mathcal{D}_0 : Res \to \mathbb{T}$$

é definida por

$$\mathcal{D}_0(r) = \mathcal{D}_1(r)(\rho_D)$$

com $\rho_D = fix(\Psi_D)$. □

Podemos, finalmente, definir a semântica denotacional para $\mathcal{L}_{syn}$

Definição 3.3.9 (Semântica denotacional)

$$\mathcal{D} : \mathcal{L}_{syn} \to \mathbb{T}$$

é definida por

$$\mathcal{D}(s) = \mathcal{D}_0(s).$$

□

Exemplos 3.3.10

1. $\mathcal{D}(a) = \mathcal{D}_1(a)(\rho_D) = \overrightarrow{a} = (\overrightarrow{a})_{n\geq 0}$ com $\overrightarrow{a}\ [0] = \emptyset$, $\overrightarrow{a}\ [1] = \{(a, \emptyset)\}$, $\overrightarrow{a}\ [n+2] = \{(a, p_\varepsilon)\}$

2. $$\begin{aligned}\mathcal{D}((a;c) \setminus c) &= \mathcal{D}_1((a;c) \setminus c)(\rho_D)\\ &= \oslash^c \mathcal{D}_1(a;c)(\rho_D)\\ &= \oslash^c \overrightarrow{x}\\ &= \overrightarrow{w}\end{aligned}$$

   onde

   $$\begin{aligned}\overrightarrow{x}\ [0] &= \emptyset & \overrightarrow{w}\ [0] &= \emptyset\\ \overrightarrow{x}\ [1] &= \{(a, \emptyset)\} & \overrightarrow{w}\ [n+1] &= \{(a, \emptyset)\}.\\ \overrightarrow{x}\ [2] &= \{(a, \{(c, \emptyset)\})\}\\ \overrightarrow{x}\ [n+3] &= \{(a, \{(c, p_\varepsilon)\})\};\end{aligned}$$

□

Apêndice: Construção do ambiente ρ_D associado à declaração D

O ambiente ρ_D irá ser construído como o ponto fixo de determinada função. O cálculo do ponto fixo irá ser feito na categoria dos cfe's com as funções conservadoras, Cfe. Comecemos por notar que o conjunto $\mathcal{U}$ (e por conseguinte $\mathbb{T}$) pode ser encarado como um cfe separado (mas não completo) com a relação de equivalência $\equiv_n$ definida por

$$\forall_{u,v\in\mathcal{U}}\ \ u \equiv_n v \Leftrightarrow u_n = v_n.$$

Com esta estrutura de cfe, $\mathcal{U}$ é solução da equação

$$\mathcal{U} \cong \{p_\varepsilon\} + \mathcal{P}_{co}(A \times \mathcal{U}^\circ)$$

sendo portanto isomorfo a $\mathbb{P}_{\mathcal{D}}$ (ver secção 3.3.2) se considerarmos o conjunto de acções Act. O primeiro passo é verificar que os operadores definidos na secção 3.3.4 são conservadores, com o conjunto $\mathbb{T}_a = \{*, \triangledown\}$ visto como um cfe discreto.

Lema 3.3.11 *Os operadores assim definidos são conservadores.*

Demonstração Vamos ver em pormenor apenas o caso da soma, os restantes casos demonstram-se de modo análogo.

Pretende-se verificar que se $u \equiv_n s$ e $v \equiv_n t$ então $u \oplus v \equiv_n s \oplus t$. Para $n = 0$ o resultado é trivial. Para $n \geq 1$ temos duas situações a considerar:

Caso $u \oplus v = \vec{p_\varepsilon}$ tem-se $\iota + \iota(u, v) = p_\varepsilon$ pelo que ou $\iota(u) = \iota(v) = p_\varepsilon$ ou $u = \triangledown$ e $\iota(v) = p_\varepsilon$ ou vice-versa. Como $u \equiv_n s$ e $v \equiv_n t$, no primeiro caso tem-se também $\iota(s) = \iota(t) = p_\varepsilon$ e no segundo caso $s = \triangledown$ e $\iota(t) = p_\varepsilon$ ou vice-versa. Em qualquer das situações vem $\iota + \iota(s, t) = p_\varepsilon$ e logo $s \oplus t = \vec{p_\varepsilon}$, como pretendido.

Caso $u \oplus v = ((u \oplus v)[k])_{k \geq 0}$, com $(u \oplus v)[0] = \emptyset$, $(u \oplus v)[k+1] = \{(a, (u' \oplus v')[k]) : (a, (u', v') \in \iota + \iota(u, v)\}$, temos de mostrar que $(u \oplus v)[n] = (s \oplus t)[n]$, ou seja, que $\{(a, (u' \oplus v')[n-1]) : (a, (u', v')) \in \iota + \iota(u, v)\} = \{(a, (s' \oplus t')[n-1]) : (a, (s', t')) \in \iota + \iota(s, t)\}$. Ora, pela definição de $\iota + \iota$, se $(a, (u', v')) \in \iota + \iota(u, v)$ estamos num dos seguintes casos:

(i) $(a, u') \in \iota(u)$ e $v' = \triangledown$;

(ii) $(a, v') \in \iota(v)$ e $u' = \triangledown$.

Sem perda de generalidade vamos supor que estamos no primeiro caso, e portanto $(a, u'[n-1]) \in u[n]$. Por outro lado, como $u \equiv_n s \Leftrightarrow u[n] = s[n]$, logo $(a, u'[n-1]) \in s[n]$ o que implica que $(a, s') \in \iota(s)$ com $s'[n-1] =$

$u'[n-1] \Leftrightarrow s' \equiv_n u'$, donde, por hipótese de indução, $(s' \oplus \nabla)[n-1] = (u' \oplus \nabla)[n-1]$ e, portanto, pela definição de ι, $(a, (s', \nabla)) \in \iota + \iota(s,t)$ logo $(a, (s' \oplus \nabla)[n-1]) \in s \oplus t[n]$ o que, atendendo ao que vimos, é o mesmo que $(a, (u' \oplus v')[n-1]) \in s \oplus t[n]$ (note-se que neste caso $v' = \nabla$). Do mesmo modo se mostrava que todo o $(a, (s' \oplus t')[n-1])$ pertencente a $s \oplus t[n]$ também pertence a $u \oplus v[n]$. □

O passo seguinte é estender os operadores de $\mathbb{T}$ a $\mathcal{U} = \mathbb{P}_{\mathcal{D}}$. Isto é feito pelo processo bem conhecido de extensão por continuidade, dado que $\mathcal{U}$ é o fecho topológico de $\mathbb{T}$.

Considerando um ambiente auxiliar $Env_{aux} = PVar \to \mathbb{P}_{\mathcal{D}}$, que é completo e separado porque $PVar$ e $\mathbb{P}_{\mathcal{D}}$ o são, calculamos o ponto fixo de $\Psi_D : Env_{aux} \to Env_{aux}$ que, em princípio, seria $\rho_D : PVar \to \mathbb{P}_{\mathcal{D}}$ mas iremos provar que $\forall x,\ \rho_D(x)$ é finito por isso $\rho_D(x) \in \mathbb{T}$.

Lema 3.3.12 *Seja* $(\rho \in) Env_{aux} = PVar \to \mathbb{P}_{\mathcal{D}}$ *tal que* $Env_{aux}(x) = Env_{aux}(x)$ *para todo o* $x \in PVar$ *e seja* $\mathcal{D}_{1aux} : Res \to Env_{aux} \to \mathbb{P}_{\mathcal{D}}$ *tal que* $\mathcal{D}_{1aux}(r) = \mathcal{D}_1(r)$ *para todo o* $r \in Res$.

Define-se $\Psi_D : Env_{aux} \to Env_{aux}$ *por* $\Psi_D(\rho)(x) = \mathcal{D}_{1aux}(D(x))(\rho)$. *Então*

(a) $\Psi_D(\rho)(x)$ *está bem definida.*

(b) Ψ_D *é aproximante em* ρ.

Demonstração É feita do mesmo modo que na semântica com $\mathsf{cfe}'s$. Vamos ver apenas alguns casos de (b).

Pretende-se mostrar que dados dois ambientes θ e ρ, se $\rho \equiv_n \theta$ então $\Psi_D(\rho) \equiv_{n+1} \Psi_D(\theta)$. Tem-se que

$$\Psi_D(\rho) \equiv_{n+1} \Psi_D(\theta) \Leftrightarrow \forall x \in PVar \Psi_D(\rho)(x) \equiv_{n+1} \Psi_D(\theta)(x)$$

$$\Leftrightarrow \forall x \in PVar, \mathcal{D}_1(D(x))(\rho) \equiv_{n+1} \mathcal{D}_1(D(x))(\theta). \quad (1)$$

Ora, $\rho \equiv_n \theta \Leftrightarrow \forall x \in PVar, \rho(x) \equiv_n \theta(x)$.

Supondo verdadeira a afirmação

$$(2)\forall x \in PVar, \rho(x) \equiv_n \theta(x) \Rightarrow \begin{cases} \forall s \in Stat, \mathcal{D}_1(s)(\rho) \equiv_n \mathcal{D}_1(s)(\theta) \\ \forall g \in GStat, \mathcal{D}_1(g)(\rho) \equiv_{n+1} \mathcal{D}_1(g)(\theta) \end{cases}$$

tem-se

$$\forall x \in PVar, \rho(x) \equiv_n \theta(x) \Rightarrow \forall x \in PVar, \mathcal{D}_1(D(x))(\rho) \equiv_{n+1} \mathcal{D}_1(D(x))(\theta),$$

visto que $D(x) \in GStat$ o que, atendendo a (1), demonstra o resultado pretendido.

Falta apenas a demonstração de (2) que, tal como no caso dos cfe's, é feita por indução estrutural em s. Vamos ver apenas o caso da soma e da restrição.

- Caso $s = s_1 + s_2$: Como $\oplus$ é conservador vem

$$\begin{array}{rcl} \mathcal{D}_1(s_1 + s_2)(\rho) & \stackrel{def.\mathcal{D}_1}{=} & \mathcal{D}_1(s_1)(\rho) \oplus \mathcal{D}_1(s_2)(\rho) \\ & \equiv_k & \mathcal{D}_1(s_1)(\theta) \oplus \mathcal{D}_1(s_2)(\theta) \\ & = & \mathcal{D}_1(s_1 + s_2)(\theta) \end{array}$$

  com $k = n + 1$ se s_1 e s_2 forem guardados e $k = n$ caso contrário.

- Caso $s = s_1 \setminus c$:

$$\begin{array}{rcl} \mathcal{D}_1(s_1 \setminus c)(\rho) & \stackrel{def.\mathcal{D}_1}{=} & \mathcal{D}_1(s_1)(\rho) \oslash c \\ & \equiv_k & \mathcal{D}_1(s_1)(\theta) \oslash c \\ & = & \mathcal{D}_1(s_1 \setminus c)(\theta) \end{array}$$

  onde, tal como anteriormente, $k = n + 1$ se s_1 e s_2 forem guardados e $k = n$ caso contrário. □

Como Ψ_D é aproximante, sabemos pelo teorema 2.1.18 que tem um ponto fixo, o ambiente em que estamos interessados, que será denotado por ρ_D. Vamos então mostrar que este ponto fixo pertence a $\mathbb{T}$ verificando que é um elemento finito.

Lema 3.3.13 *Fazendo $\rho_D = fix(\Psi_D)$ tem-se*

$$\forall x \in PVar,\ \rho_D(x) \in \mathbb{T}$$

Demonstração Sabemos que $\rho_D = fix(\Psi_D)$, logo

$$\forall x \in PVar\ \ \rho_D(x) = \Psi_D(\rho_D)(x) = \mathcal{D}_{1aux}(D(x))(\rho_D)$$

Temos de verificar que $\forall x \in PVar\ \ \mathcal{D}_{1aux}(D(x))(\rho_D)$ é finito. Vamos demonstrar por indução estrutural em $D(x)(\in GStat)$

- caso $D(x) = a$: $\mathcal{D}_{1aux}(a)(\rho_D) = \vec{a}$, logo é finito;
- caso $D(x) = g \setminus c$: $\mathcal{D}_{1aux}(g \setminus c)(\rho_D) = \oslash^c \mathcal{D}_1(g)(\rho_D)$, onde por hipótese de indução $\mathcal{D}_1(g)(\rho_D)$ é finito, logo, pela definição de $\oslash^c$ tem-se $\oslash^c \mathcal{D}_1(g)(\rho_D)$ finito.

Os restantes casos demonstram-se de modo análogo. □

3.3.5 Equivalência entre a semântica operacional e a semântica denotacional

Tal como acontecia no caso das semânticas com cfe's, não é possível estabelecer uma equivalência directa entre a semântica operacional e a semântica denotacional. Temos de aplicar primeiro a $\mathcal{D}$ uma função que elimine os ramos a partir do ponto em que contenham acções de sincronização.

Definição 3.3.14 (Função de abstracção) Seja $abs : \mathbb{T}_{\mathcal{D}} \to \mathbb{T}_{\mathcal{O}}$ definida por $abs(\vec{t}) = (abs_n(\vec{t}_n))_{n\geq 0}$ onde

$abs_0(\vec{t}_0) = \emptyset$

$$abs_{n+1}(\vec{t}_{n+1}) = \begin{cases} p_\varepsilon & \text{se } \vec{t}_{n+1} = p_\varepsilon, \\ \{(b, abs_n(u) : (b,u) \in \vec{t}_{n+1}, b \in IAct\} & \text{caso contrário.} \end{cases}$$

□

Vamos precisar ainda de alguns resultados auxiliares que apresentamos de seguida.

Lema 3.3.15 $u \oplus \nabla = \nabla \oplus u = u$

Demonstração Se $u \oplus \nabla = \vec{p_\varepsilon}$, pela definição de $\oplus$, é porque $\iota + \iota(u, \nabla) = p_\varepsilon$, logo $\iota(u) = p_\varepsilon$ e será, necessariamente, $u = \vec{p_\varepsilon}$, como se pretende.

Caso contrário, $u \oplus \nabla = \vec{z}$ com

$$\begin{aligned} \vec{z}\,[0] = \emptyset &= u[0] \\ \vec{z}\,[n+1] &= \{(a, (u' \oplus v')[n]) : (a,(u',v')) \in \iota + \iota(u,\nabla)\} \\ &= (\text{por definição de } \iota + \iota) \\ &\quad \{(a, (u' \oplus \nabla)[n]) : (a,u') \in \iota(u)\} \\ &= (\text{por hipótese de indução em } n) \\ &\quad \{(a, u'[n]) : (a,u') \in \iota(u)\} \\ &= u[n+1], \end{aligned}$$

atendendo a que $\iota(u) = \{(a,u') : \forall_n\ (a, u'[n]) \in u[n+1]\}$.

O caso $\nabla \oplus u$ demonstra-se de forma análoga. □

Lema 3.3.16 $\iota(\mathcal{D}(s)) = \{(a, \mathcal{D}(r)) : s \xrightarrow{a} r\}$

Demonstração Vamos fazer a demonstração por indução em peso.

- Caso $s = a$,

$$\begin{aligned}
\iota(\mathcal{D}(a)) &= \{(a,v) : \forall_n\ (a, v[n]) \in \mathcal{D}_1(a)(\rho_D)[n+1]\} \\
&= (\text{por definição de } \mathcal{D}_1) \\
&\quad \{(a,v) : \forall_n\ (a, v[n]) \in \overrightarrow{a}\ [n+1]\} \\
&= (\text{por definição de } \overrightarrow{a}) \\
&\quad \{(a, \overrightarrow{p_\varepsilon})\} \\
&= (\text{por definição de } \mathcal{D}) \\
&\quad \{(a, \mathcal{D}(E))\} \\
&= (\text{por definição de } \mathcal{T}_{syn}) \\
&\quad \{(a, \mathcal{D}(r)) : a \xrightarrow{a} r\}.
\end{aligned}$$

- Caso $s = x$, da definição de $\mathcal{T}_{syn}$ sabemos que

$$\begin{aligned}
\iota(\mathcal{D}(x)) &= \{(a,v) : \forall_n (a, v[n]) \in \mathcal{D}_1(x)(\rho_D)[n+1]\} \\
&= (\text{por definição de } \mathcal{D}) \\
&\quad \{(a,v) : \forall_n (a, v[n]) \in \rho_D(x)[n+1]\} \\
&= (\text{porque } \rho_D \text{ é ponto fixo } \Psi_D) \\
&\quad \{(a,v) : \forall_n (a, v[n]) \in \mathcal{D}_1(D(x))(\rho_D)[n+1]\} \\
&= (\text{por definição de } \iota \text{ e } \mathcal{D}) \\
&\quad \iota(\mathcal{D}(D(x))) \\
&= (\text{por hipótese de indução em } \mathsf{peso}) \\
&\quad \{(a, \mathcal{D}(r)) : D(x) \xrightarrow{a} r\} \\
&= (\text{por definição de } \mathcal{T}_{syn}) \\
&\quad \{(a, \mathcal{D}(r)) : x \xrightarrow{a} r\}.
\end{aligned}$$

$$\begin{aligned}
\iota(\mathcal{D}(s_1 + s_2)) &= \{(a,v) : \forall_n (a, v[n]) \in \mathcal{D}(s_1 + s_2)[n+1]\} \\
&= (\text{por definição de } \mathcal{D}) \\
&\quad \{(a,v) : \forall_n (a, v[n]) \in (\mathcal{D}(s_1) \oplus \mathcal{D}(s_2))[n+1]\} \\
&= (\text{por definição de } \oplus) \\
&\quad \{(a,v) : \forall_n (a, v[n]) \in \{(a, (x \oplus y)[n]) : \\
&\qquad\qquad (a, (x,y)) \in \iota + \iota(\mathcal{D}(s_1), \mathcal{D}(s_2))\}\}
\end{aligned}$$

$$
\begin{array}{rl}
= & (\text{por definição de } \iota + \iota) \\
& \{(a,v) : \forall_n (a, v[n]) \in \\
& \qquad (\{(a_1, (x \oplus \nabla)[n]) : (a_1, x) \in \iota(\mathcal{D}(s_1))\} \\
& \qquad \cup \{(a_2, (\nabla \oplus y)[n]) : (a_2, y) \in \iota(\mathcal{D}(s_2))\})\} \\
= & (\text{por hipótese de indução em } \mathsf{peso}) \\
& \{(a,v) : \forall_n (a, v[n]) \in (\{(a_1, (\mathcal{D}(r_1) \oplus \nabla)[n]) : s_1 \xrightarrow{a_1} r_1\} \\
& \qquad \cup \{(a_2, (\nabla \oplus \mathcal{D}(r_2))[n]) : s_2 \xrightarrow{a_2} r_2\})\} \\
= & (\text{pelo lema 3.3.15}) \\
& \{(a,v) : \forall_n (a, v[n]) \in (\{(a_1, \mathcal{D}(r_1)[n]) : s_1 \xrightarrow{a_1} r_1\} \\
& \qquad \cup \{(a_2, \mathcal{D}(r_2)[n]) : s_2 \xrightarrow{a_2} r_2\})\} \\
= & (\text{porque } v \in \mathbb{T}) \\
& \{(a_1, \mathcal{D}(r_1)) : s_1 \xrightarrow{a_1} r_1\} \\
& \qquad \cup \{(a_2, \mathcal{D}(r_2)) : s_2 \xrightarrow{a_2} r_2\} \\
= & (\text{por definição de } \mathcal{T}_{syn}) \\
& \{(a, \mathcal{D}(r)) : s_1 + s_2 \xrightarrow{a} r\}
\end{array}
$$

- Caso $s = s_1 \parallel s_2$, temos

$$
\begin{array}{rcl}
\iota(\mathcal{D}(s_1 \parallel s_2)) & = & \{(a,v) : \forall_n (a, v[n]) \in \mathcal{D}(s_1 \parallel s_2)[n+1]\} \\
& = & (\text{por definição de } \mathcal{D}) \\
& & \{(a,v) : \forall_n (a, v[n]) \in (\mathcal{D}(s_1) \between \mathcal{D}(s_2))[n+1]\} \\
& = & (\text{por definição de } \between) \\
& & \{(a,v) : \forall_n (a, v[n]) \in \\
& & \qquad \{(a, (x \between y)[n]) : (a, (x,y)) \in \iota \parallel \iota(\mathcal{D}(s_1), \mathcal{D}(s_2))\}\} \\
& = & (\text{por definição de } \iota \between \iota) \\
& & \{(a,v) : \forall_n (a, v[n]) \in \\
& & \qquad (\{(\tau, (x \between y)[n]) : (c, x) \in \iota(\mathcal{D}(s_1)), (\overline{c}, x) \in \iota(\mathcal{D}(s_2))\} \\
& & \qquad \cup \{(a_1, (x \between \mathcal{D}(s_2))[n]) : (a_1, x) \in \iota(\mathcal{D}(s_1))\} \\
& & \qquad \cup \{(a_2, (\mathcal{D}(s_1) \between y)[n]) : (a_2, x) \in \iota(\mathcal{D}(s_2))\})\}
\end{array}
$$

$$
\begin{aligned}
= \quad & (\text{por hipótese de indução em } \mathsf{peso}) \\
& \{(a,v) : \forall_n (a, v[n]) \in \\
& \qquad (\{(\tau, (\mathcal{D}(r_1) \textcircled{\scriptsize ||} \mathcal{D}(r_2))[n]) : s_1 \xrightarrow{c} r_1, s_2 \xrightarrow{\overline{c}} r_2\} \\
& \qquad \cup \{(a_1, (\mathcal{D}(r_1) \textcircled{\scriptsize ||} \mathcal{D}(s_2))[n]) : s_1 \xrightarrow{a_1} r_1\} \\
& \qquad \cup \{(a_2, (\mathcal{D}(s_1) \textcircled{\scriptsize ||} \mathcal{D}(r_2))[n]) : s_2 \xrightarrow{a_2} r_2\})\} \\
= \quad & (\text{porque } v \in \mathbb{T}) \\
& \{(\tau, \mathcal{D}(r_1) \textcircled{\scriptsize ||} \mathcal{D}(r_2)) : s_1 \xrightarrow{c} r_1, s_2 \xrightarrow{\overline{c}} r_2\} \\
& \cup \{(a_1, \mathcal{D}(r_1) \textcircled{\scriptsize ||} \mathcal{D}(s_2)) : s_1 \xrightarrow{a_1} r_1\} \\
& \cup \{(a_2, \mathcal{D}(s_1) \textcircled{\scriptsize ||} \mathcal{D}(r_2)) : s_2 \xrightarrow{a_2} r_2\} \\
= \quad & (\text{por definição de } \mathcal{D} \text{ e } \mathcal{T}_{syn}) \\
& \{(\tau, \mathcal{D}(r_1 \parallel r_2)) : s_1 \xrightarrow{c} r_1, s_2 \xrightarrow{\overline{c}} r_2\} \\
& \cup \{(a_1, \mathcal{D}(r_1 \parallel s_2)) : s_1 \xrightarrow{a_1} r_1\} \\
& \cup \{(a_2, \mathcal{D}(s_1 \parallel r_2)) : s_2 \xrightarrow{a_2} r_2\} \\
= \quad & (\text{por definição de } \mathcal{T}_{syn}) \\
& \{(a, \mathcal{D}(r)) : s_1 \parallel s_2 \xrightarrow{a} r\}
\end{aligned}
$$

Os casos em que $s = s_1; s_2$ e $s = s_1 \setminus c$ demonstram-se de modo análogo. □

Teorema 3.3.17

$$\mathcal{O} = abs \circ \mathcal{D}$$

Demonstração Temos de mostrar que, para todo o $r \in Res$ tem-se $\mathcal{O}(r) = abs \circ \mathcal{D}_0(r)$. Vamos fazer a demonstração por indução em peso:

Caso $r = E$, $\mathcal{O}(E) = \overrightarrow{p_\varepsilon} = abs \circ \mathcal{D}_1(E)(\rho_D) = \mathcal{D}_0(E)$.

Caso $r = a$, temos de considerar duas situações distintas:

Se $a = b (\in IAct)$ vem $\mathcal{O}(b) = \overrightarrow{b} = abs(\overrightarrow{b}) = abs \circ \mathcal{D}_1(b)(\rho_D) = abs \circ \mathcal{D}(b)$.

Se $a = c (\in Sync)$ vem $\mathcal{O}(c) = \overrightarrow{\emptyset} = abs(\overrightarrow{c}) = abs \circ \mathcal{D}_1(c)(\rho_D) = abs \circ \mathcal{D}(c)$.

Caso $r = x$, temos

$$\begin{aligned}
\mathcal{O}(x) &= \mathcal{O}(D(x)) \\
&= \text{(por hipótese de indução)} \\
&\quad abs \circ \mathcal{D}(D(x)) \\
&= \text{(por definição de } \mathcal{D}) \\
&\quad abs \circ \mathcal{D}_1(D(x))(\rho_D) \\
&= \text{(porque } \rho_D \text{ é ponto fixo de } \Psi_D) \\
&\quad abs \circ \rho_D(x) \\
&= \text{(por definição } \mathcal{D}_1) \\
&\quad abs \circ \mathcal{D}_1(x)(\rho_D) \\
&= \text{(por definição de } \mathcal{D}) \\
&\quad abs \circ \mathcal{D}(x).
\end{aligned}$$

Caso $r = s_1 + s_2$, temos de verificar que para todo o n se tem

$$\mathcal{O}(s_1 + s_2)[n] = abs \circ \mathcal{D}(s_1 + s_2)[n].$$

Para $n = 0$ vem, $\mathcal{O}(s_1 + s_2)[0] = \emptyset = abs \circ \mathcal{D}(s_1 + s_2)[0]$.

Para $n + 1$ vem,

$$\begin{aligned}
\mathcal{O}(s_1 + s_2)[n+1] &= \{(b, \mathcal{O}(r)[n]) : s_1 + s_2 \xrightarrow{b} r\} \\
&= \text{(por definição de } \mathcal{T}_{syn}) \\
&\quad \{(b_1, \mathcal{O}(r_1)[n]) : s_1 \xrightarrow{b_1} r_1\} \\
&\quad \cup \{(b_2, \mathcal{O}(r_2)[n]) : s_2 \xrightarrow{b_2} r_2\} \\
&= \text{(por hipótese de indução em } \mathsf{peso}) \\
&\quad \{(b_1, abs_n \circ \mathcal{D}(r_1)[n]) : s_1 \xrightarrow{b_1} r_1\} \\
&\quad \cup \{(b_2, abs_n \circ \mathcal{D}(r_2)[n]) : s_2 \xrightarrow{b_2} r_2\}.
\end{aligned}$$

Por outro lado, tem-se,

$$
\begin{array}{rcl}
(abs \circ \mathcal{D}(s_1 + s_2))[n+1] & = & abs_{n+1} \circ \mathcal{D}(s_1 + s_2)[n+1] \\
& = & \text{(por definição de } \mathcal{D}) \\
& & abs_{n+1} \circ \mathcal{D}_1(s_1 + s_2)(\rho_D)[n+1] \\
& = & \text{(por definição de } \mathcal{D}_1) \\
& & abs_{n+1} \circ (\mathcal{D}_1(s_1)(\rho_D) \oplus \mathcal{D}_1(s_2)(\rho_D))[n+1] \\
& = & \text{(por definição de } \oplus) \\
& & abs_{n+1} \circ (\{(a, (u' \oplus v')[n] : \\
& & \qquad (a, (u', v')) \in \iota + \iota(\mathcal{D}_1(s_1)(\rho_D), \mathcal{D}_1(s_2)(\rho_D))\} \\
& = & \text{(por definição de } \iota + \iota) \\
& & abs_{n+1} \circ (\{(a_1, (u \oplus \nabla)[n] : (a_1, u) \in \iota(\mathcal{D}_1(s_1)(\rho_D))\} \\
& & \qquad \cup \{(a_2, (\nabla \oplus v)[n] : (a_2, u) \in \iota(\mathcal{D}_1(s_2)(\rho_D))\}) \\
& = & \text{(pelo lema 3.3.15)} \\
& & abs_{n+1} \circ (\{(a_1, u[n] : (a_1, u) \in \iota(\mathcal{D}_1(s_1)(\rho_D))\} \\
& & \qquad \cup \{(a_2, v[n] : (a_2, u) \in \iota(\mathcal{D}_1(s_2)(\rho_D))\}) \\
& = & \text{(por definição de } \mathcal{D}) \\
& & abs_{n+1} \circ (\{(a_1, u[n] : (a_1, u) \in \iota(\mathcal{D}(s_1))\} \\
& & \qquad \cup \{(a_2, v[n] : (a_2, u) \in \iota(\mathcal{D}(s_2))\}) \\
& = & \text{(por definição de } abs) \\
& & \{(b_1, abs_n(u[n]) : (b_1, u) \in \iota(\mathcal{D}(s_1))\} \\
& & \qquad \cup \{(b_2, abs_n(v[n])) : (b_2, u) \in \iota(\mathcal{D}(s_2))\} \\
& = & \text{(pelo lema 3.3.16)} \\
& & \{(b_1, abs_n(\mathcal{D}(r_1)[n]) : s_1 \xrightarrow{b_1} r_1\} \\
& & \qquad \cup \{(b_2, abs_n(\mathcal{D}(r_2)[n])) : s_2 \xrightarrow{b_2} r_2\},
\end{array}
$$

como pretendíamos.

Caso $r = s_1; s_2$, temos de verificar que para todo o n se tem

$$\mathcal{O}(s_1; s_2)[n] = abs \circ \mathcal{D}(s_1; s_2)[n].$$

Para $n = 0$ é trivial.

Para $n + 1$ tem-se, por um lado,

$$
\begin{array}{rcl}
\mathcal{O}(s_1; s_2)[n+1] & = & \{(b, \mathcal{O}(r)[n]) : s_1; s_2 \xrightarrow{b} r\} \\
& = & \text{(por definição de } \mathcal{T}_{syn}) \\
& & \{(b, \mathcal{O}(r_1; s_2)[n]) : s_1 \xrightarrow{b} r_1\} \\
& = & \text{(por hipótese de indução em } n) \\
& & \{(b, (abs \circ \mathcal{D}(r_1; s_2))[n]) : s_1 \xrightarrow{b} r_1\} \\
& = & \text{(por definição de } \mathcal{D}, abs) \\
& & \{(b, abs_n(\mathcal{D}_1(r_1; s_2)(\rho_D)[n])) : s_1 \xrightarrow{b} r_1\} \\
& = & \text{(por definição de } \mathcal{D}_1) \\
& & \{(b, abs_n(\mathcal{D}_1(r_1)(\rho_D) \dot{\cup} \mathcal{D}_1(s_2)(\rho_D))[n]) : s_1 \xrightarrow{b} r_1\} \\
& = & \text{(por definição de } \mathcal{D}) \\
& & \{(b, abs_n(\mathcal{D}(r_1) \dot{\cup} \mathcal{D}(s_2))[n])) : s_1 \xrightarrow{b} r_1\}.
\end{array}
$$

Por outro lado,

$$
\begin{array}{rcl}
(abs \circ \mathcal{D}(s_1; s_2))[n+1] & = & abs_{n+1}(\mathcal{D}(s_1; s_2)[n+1]) \\
& = & \text{(por definição de } \mathcal{D}) \\
& & abs_{n+1}(\mathcal{D}_1(s_1; s_2)(\rho_D)[n+1]) \\
& = & \text{(por definição de } \mathcal{D}_1) \\
& & abs_{n+1}((\mathcal{D}_1(s_1)(\rho_D) \dot{\cup} \mathcal{D}_1(s_2)(\rho_D))[n+1]) \\
& = & \text{(por definição de } \mathcal{D}) \\
& & abs_{n+1}((\mathcal{D}(s_1) \dot{\cup} \mathcal{D}(s_2))[n+1]) \\
& = & \text{(por definição de } \dot{\cup}) \\
& & abs_{n+1}(\{(a, (u \dot{\cup} v)[n]) : (a, (u, v)) \in \iota; \iota(\mathcal{D}(s_1), \mathcal{D}(s_2))\}) \\
& = & \text{(por definição de } \iota; \iota) \\
& & abs_{n+1}(\{(a, (u' \dot{\cup} \mathcal{D}(s_2))[n]) : (a, u') \in \iota(\mathcal{D}(s_1))\}) \\
& = & \text{(pelo lema 3.3.16)} \\
& & abs_{n+1}(\{(a, (\mathcal{D}(r_1) \dot{\cup} \mathcal{D}(s_2)[n]) : s_1 \xrightarrow{a} r_1\} \\
& = & \text{(por definição de } abs) \\
& & \{(b, abs_n(\mathcal{D}(r_1) \dot{\cup} \mathcal{D}(s_2))[n])) : s_1 \xrightarrow{b} r_1\}
\end{array}
$$

como pretendíamos. □

3.4 Comparação entre as semânticas definidas utilizando cfe's e as definidas utilizando coálgebras

Nesta secção vamos estabelecer duas equivalências entre as semânticas obtidas utilizando cfe's e as semânticas obtidas utilizando coálgebras. Vamos utilizar o índice $_{\mathsf{cfe}}$ sempre que nos referirmos às funções obtidas pela técnica dos cfe's e o índice $_{coal}$ quando nos referirmos às funções obtidas pela técnica das coálgebras.

O primeiro resultado é um teorema que estabelece a igualdade entre as semânticas operacionais obtidas por cada um dos processos.

Teorema 3.4.1

$$\mathcal{O}_{\mathsf{cfe}} = \mathcal{O}_{coal}$$

Demonstração A primeira função semântica operacional que definimos, $\mathcal{O}_{\mathsf{cfe}}[\![\,.\,]\!] : \mathcal{L}_{syn} \to \mathbb{P}$, é dada por $\mathcal{O}_{\mathsf{cfe}}[\![\,s\,]\!] = \mathcal{O}_{\mathsf{cfe}}(s)$; e a segunda função $\mathcal{O}_{coal}[\![\,.\,]\!] : \mathcal{L}_{syn} \to \mathbb{T}$ é dada por $\mathcal{O}[\![\,s\,]\!] = \mathcal{O}(s)$. Temos de mostrar que dados $\mathcal{O}_{\mathsf{cfe}} : \mathcal{L}_{syn} \to \mathbb{P}$ e $\mathcal{O}_{coal} : \mathcal{L}_{syn} \to \mathbb{T}$ se tem, para todo o $r \in Res$, $\mathcal{O}_{\mathsf{cfe}}(r) = \mathcal{O}_{coal}(r)$. Ora isto é imediato pois no caso $r = E$ temos $\mathcal{O}_{coal}(E) = \overrightarrow{p_\varepsilon} = \mathcal{O}_{\mathsf{cfe}}(E)$ e para o caso $r = s$ temos

$$\mathcal{O}_{coal}(s)[0] = \emptyset = \mathcal{O}_{\mathsf{cfe}}(s)[0].$$

e para n+1

$$\begin{aligned}\mathcal{O}_{coal}(s)[n+1] &= \{(b, \mathcal{O}_{coal}(r)[n]) : s \xrightarrow{b} r\} \\ &= \{(b, \mathcal{O}_{\mathsf{cfe}}(r)[n]) : s \xrightarrow{b} r\} \\ &= \mathcal{O}_{\mathsf{cfe}}(s)[n+1],\end{aligned}$$

onde a segunda igualdade é obtida por hipótese de indução em n. □

Em seguida apresentamos um resultado que relaciona as semânticas denotacionais obtidas por cada um dos processos. Trata-se de uma consequência quase imediata do teorema anterior e das relações estabelecidas entre as semânticas operacional e denotacional nas duas secções anteriores.

Corolário 3.4.2

$$abs_{\mathsf{cfe}} \circ \mathcal{D}_{\mathsf{cfe}} = abs_{coal} \circ \mathcal{D}_{coal}$$

Demonstração Sabemos, pelo teorema 3.2.16, que $\mathcal{O}_{\mathsf{cfe}} = abs_{\mathsf{cfe}} \circ \mathcal{D}_{\mathsf{cfe}}$ e, pelo teorema 3.3.17, que $\mathcal{O}_{coal} = abs_{coal} \circ \mathcal{D}_{coal}$, pelo que atendendo ao resultado do teorema 3.4.1, o resultado é imediato. □

Na definição com base em cfe's, embora os métodos utilizados sejam muito semelhantes aos das técnicas métricas, evitam-se algumas construções de ponto fixo e as demonstrações são quase todas feitas por indução o que simplifica muitos dos processos.

É interessante verificar que ambas as semânticas operacionais são obtidas directamente do sistema de transições sem ser necessário recorrer a resultados de ponto fixo.

Na definição coalgébrica das semânticas mostrou-se como os operadores semânticos podem ser definidos coalgebricamente em vez de o serem por utilização de técnicas de ponto fixo. Os domínios utilizados com esta técnica são mais intuitivos porque usam as potências finitas ($\mathcal{P}_{fin}$) em vez das potências compactas ($\mathcal{P}_{co}$) usadas na abordagem com cfe's.

Para a semântica denotacional não fizemos uma abordagem coalgébrica pura, introduzimos uma técnica métrica, para o cálculo do ambiente ρ_D associado à declaração D, embora estejamos convencidos que seria seria possível fazer uma abordagem pura. É um assunto que podemos vir a explorar de futuro.

Capítulo 4

Semântica de uma linguagem com mobilidade

A mobilidade é um conceito chave na computação distribuída e nas linguagens concorrentes orientadas para objectos. Podemos distinguir dois tipos diferentes de mobilidade. Num, designado por paradigma de primeira ordem (ou passagem de nomes), são as ligações entre os processos que são alteradas dinamicamente num espaço abstracto de processos ligados entre si, como exemplos temos as ligações de hipertexto que podem ser criadas, passadas a outro e eliminadas; as ligações entre telefones celulares e a rede de estações base que vai-se alterando à medida que o telefone é levado de um local para outro; nos sistemas orientados para objectos, as referências podem ser passadas como argumentos na invocação de um método. No outro, designado por paradigma de ordem superior (ou passagem de processos), são os processos que se movem num espaço abstracto de processos ligados entre si. Por exemplo, um bloco de código pode ser enviado através de uma rede e ser corrido no seu destino; nos sistemas orientados para objectos, um procedimento pode ser passado como um argumento na invocação de um método. Nesta categoria estão o cálculo-γ e o CLM, entre outros. Tanto conceptualmente como matematicamente a passagem de nomes é mais simples que a passagem

de processos e pensa-se que a passagem de nomes poderá ser suficiente para modelar as comunicações em que há passagem de processos.

O nossa abordagem à mobilidade vai limitar-se ao paradigma de primeira ordem cujo representante principal é o cálculo-π. O cálculo-π é uma extensão da álgebra de processos CCS (Calculus of Communicating Systems) de Milner [Mil89], desenvolvida nos anos 80 por R. Milner, J. Parrow e D. Walker [MPW92], com o objectivo de expressar o comportamento de sistemas com mobilidade. Trata-se de um modelo matemático para descrever processos cujas interligações sofrem modificações à medida que os processos interagem. O cálculo-π tem a particularidade de os seus canais (ou portas) de comunicação serem identificados por nomes e as computações consistirem, simplesmente, na comunicação de nomes de canais através dos canais de ligação. Intuitivamente, os nomes representam a capacidade de aceder ao canal, um processo ao passar um nome x a outro processo é como se lhe passasse uma ligação ao canal x. Um nome (uma ligação) pode ser do conhecimento geral ou pode ser privado e ter um âmbito restrito a um ou mais processos. A possibilidade de reconfigurar as ligações (mobilidade) é alcançada pelo facto de que quando um processo transmite para o exterior um nome x que era conhecido apenas localmente, o processo que recebe esta informação adquire a capacidade de comunicar pelo canal privado x. Esta capacidade de reconfigurar as ligações entre os processos permite utilizar o cálculo-π para modelar linguagens orientadas a objectos [Wal95], e para codificar comunicações de ordem superior [San92]. Contudo esta capacidade de alterar dinamicamente o alcance dos nomes locais levanta problemas novos quando se pretende estender ao cálculo-π as técnicas desenvolvidas para álgebras de processos [Len98, AC98]. Têm surgido várias versões de semânticas denotacionais para o cálculo-π nomeadamente em [Bal00] e em [Sta96], esta última referência foi uma das que inspiraram este trabalho. O facto de os nomes serem transmitidos durante as interacções conduz ao aparecimento de duas famílias distintas de semânticas dependendo da intuição operacional sobre

as acções de entrada ([FMQ95, MPW93]). O acto de efectuar a entrada e a escolha do nome recebido podem ser considerados como um único evento atómico – perspectiva precoce (*early*) – ou como dois eventos conceptualmente diferentes – perspectiva tardia (*late*). Uma terceira abordagem é a seguida por Milner em [Mil99] com base em abstracções (abstractions) e concretizações (concretions). No nosso trabalho vamos adoptar uma perspectiva semelhante a esta última, embora menos marcadamente sintáctica e baseada num modelo coalgébrico.

Neste capítulo começamos por apresentar a sintaxe do cálculo-π, algumas definições básicas e o domínio semântico que iremos utilizar nas secções seguintes. São então definidas uma semântica operacional e uma semântica denotacional para o cálculo-π, com base em coálgebras. Prova-se a equivalência entre as duas semânticas no caso fechado. Terminamos mostrando que o núcleo de equivalência das semânticas coincide com a bissimilaridade forte, no caso fechado, e com a congruência forte no caso aberto. Para o estudo vamos considerar uma versão monádica (é passado apenas um nome em cada comunicação) e síncrona do cálculo-π, baseada na definição das expressões de processo apresentada em [Mil99], que inclui as operações de soma, composição paralela, restrição e replicação.

4.1 Sintaxe do Cálculo-π e definições básicas

Assumimos que existe um conjunto infinito enumerável de nomes, $\mathcal{N}$, que representam todas os canais de comunicação e também as variáveis e os dados a transmitir. Os elementos de $\mathcal{N}$ serão denotados por $a, b, ..., x, y, z, ...$ Os prefixos de acção π representam ou o envio de uma mensagem ou a recepção de uma mensagem ou ainda uma acção interna.

Definição 4.1.1 (Prefixos) Sejam $a, x \in \mathcal{N}$. Os prefixos de acção π definem-

se por

$$\begin{array}{rcll} \pi & ::= & \overline{a}\langle x\rangle & \text{(saida)} \\ & & a(x) & \text{(entrada)} \\ & & \tau & \text{(acção interna)} \end{array}$$

□

O prefixo de saída, $\overline{a}\langle x\rangle$, corresponde ao envio do nome x pelo canal a. O canal pode ser visto como uma porta de saída, denota-se por $\overline{a}$, e x como a informação enviada. O prefixo de entrada, $a(x)$, corresponde à recepção de um nome arbitrário, z, pelo canal a e x funciona como um parâmetro formal que será substituído pelo nome z recebido. A acção interna τ representa uma comunicação interna ao processo, correspondendo a um envio e recepção síncronos (simultâneos) de uma mensagem (nome).

Definição 4.1.2 (Processos) Seja $x \in \mathcal{N}$. O conjunto $(P \in)\mathsf{Proc}$ das expressões de processo π é definido por

$$\begin{array}{rcll} P & ::= & \mathsf{0} & \text{(nulo)} \\ & & \pi.P & \text{(prefixo)} \\ & & P + P & \text{(soma)} \\ & & P|P & \text{(composição paralela)} \\ & & (\nu x)P & \text{(restricção)} \\ & & !P & \text{(replicação)} \end{array}$$

□

O processo nulo, $\mathsf{0}$, é um processo que não pode efectuar nenhuma acção. O processo na forma prefixa, $\pi.P$, corresponde a efectuar a acção associada ao prefixo π e prosseguir como P ou, no caso de o prefixo ser de entrada, $a(x)$, após a recepção do nome, digamos z, prosseguir como o processo obtido de P por substituição do nome x pelo nome recebido, z. A soma, $P+Q$, dos processos P e Q representa um agente que pode evoluir como P ou como Q,

é, portanto, uma escolha não determinista. A soma é comutativa, associativa e tem como elemento neutro o processo nulo, propriedades estas que serão incluídas na congruência estrutural. A composição paralela, $P|Q$, consiste na execução em paralelo de P e Q. Os componentes P e Q podem actuar de forma independente ou podem comunicar entre si, sincronizando-se se um deles efectua uma saída e o outro uma entrada pelo mesmo canal de comunicação, dando origem a uma acção interna. A composição paralela também é comutativa, associativa e tem como elemento neutro o processo nulo; tal como para a soma, estas propriedades serão tidas em consideração mais adiante na definição de congruência estrutural. A replicação, $!P$, representa a capacidade de criar um número ilimitado de "cópias" de P que são executadas em composição paralela com $!P$. A replicação junto com a passagem de nomes como mensagens fornece a este cálculo todo o poder das definições paramétricas explícitas, que utilizam equações recursivas, e tem a vantagem de simplificar consideravelmente a teoria.

A ocorrência de um nome x num processo diz-se *ligada* se aparece numa das formas $(\nu x)P$ ou $a(x).P$, caso contrário diz-se *livre*. Em ambos os casos o âmbito da ocorrência ligada é o processo P. O conjunto dos nomes que ocorrem ligados num processo P denota-se por $lg(P)$ e o conjunto dos nomes que ocorrem livres num processo P denota-se por $lv(P)$.

Definição 4.1.3 (Nomes livres de um processo) Para todo o processo P, o conjunto dos nomes livres de P, denotado por $lv(P)$, define-se indutivamente por

$$
\begin{aligned}
&lv(\mathsf{0}) = \emptyset;\\
&lv(\overline{x}\langle y\rangle.P) = \{x, y\} \cup lv(P);\\
&lv(x(y).P) = \{x\} \cup (lv(P) - \{y\});\\
&lv(\tau.P) = lv(P);\\
&lv(P \,_{\mathrm{op}}\, \mathrm{Q}) = lv(P) \cup lv(Q), \;\; {}_{\mathrm{op}} \in \{+, |\};\\
&lv((\nu x)P) = lv(P) - \{x\};\\
&lv(!P) = lv(P).
\end{aligned}
$$

□

Os nomes ligados de um processo podem ser também definidos indutivamente.

Definição 4.1.4 (Nomes ligados de um processo) Para todo o processo P, o conjunto dos nomes ligados de P, denotado por $lg(P)$, define-se indutivamente por

$$
\begin{aligned}
&lg(\mathsf{0}) = \emptyset;\\
&lg(\overline{x}\langle y\rangle.P) = lg(P);\\
&lg(x(y).P) = \{y\} \cup lg(P);\\
&lg(\tau.P) = lg(P);\\
&lg(P \,\mathrm{op}\, \mathrm{Q}) = lg(P) \cup lg(Q), \;\; \mathrm{op} \in \{+, |\};\\
&lg((\nu x)P) = \{x\} \cup lg(P);\\
&lg(!P) = lg(P).
\end{aligned}
$$

□

Um nome que não ocorra livre num processo designa-se por nome *novo* ou nome *fresco*.

Chamamos *conversão-α* à técnica de, num processo, trocar um nome ligado por um nome novo. Dois processos P e Q dizem-se *equivalentes-α* e escreve-se $P \equiv_\alpha Q$ se cada um pode ser obtido do outro por conversão-α, um nome de cada vez.

Uma *substituição* é uma função $\theta : \mathcal{N} \to \mathcal{N}$ tal que $\theta(x) \neq x$ apenas para um número finito de nomes. Se $\theta(x_i) = y_i$ para $1 \leq i \leq n$ e $\theta(x) = x$ para todos os outros nomes, escrevemos θ como $\{y_1/x_1, ..., y_n/x_n\}$. Note-se que uma permutação é um caso particular de uma substituição.

A aplicação de uma substituição a um processo P, que se denota por $P\theta$ ou $P\{y_1/x_1, ..., y_n/x_n\}$, é a substituição simultânea em P de todas ocorrências livres de x_i por y_i, para $1 \leq i \leq n$, efectuando conversões-α, se necessário, para impedir que algum dos nomes y_i se torne ligado em P. Deste modo, se aplicarmos a substituição $\theta = \{y/x\}$ ao processo $P = \overline{a}\langle x\rangle.\overline{b}\langle y\rangle.\mathsf{0}$, que usualmente se representa apenas por $P = \overline{a}\langle x\rangle.\overline{b}\langle y\rangle$, obtém-se $P\theta = \overline{a}\langle y\rangle.\overline{b}\langle y\rangle$.

Mas se pretendermos aplicar a mesma substituição a $(\nu y)P$ uma vez que y ocorre agora ligado, temos de efectuar primeiro uma conversão-α de forma a que o y introduzido por θ não seja capturado pelo mecanismo de ligação (νy). Deste modo, em primeiro lugar efectuamos uma conversão-α conveniente, por exemplo $(\nu z)\overline{a}\langle x\rangle.\overline{b}\langle z\rangle$, e só depois aplicamos a substituição obtendo $((\nu z)\overline{a}\langle x\rangle.\overline{b}\langle z\rangle)\theta = (\nu z)\overline{a}\langle y\rangle.\overline{b}\langle z\rangle$. Para simplificar a escrita, sempre que não houver perigo de confusão, um processo $P.\mathsf{0}$ será denotado simplesmente por P.

À semelhança do que acontece no cálculo-λ, não queremos fazer distinção entre processos que diferem apenas por conversão-α. Por exemplo, consideramos iguais os processos $a(x).P$ e $a(z).P\{z/x\}$ onde $z \notin lv(P) - \{x\}$.

4.2 Domínio Semântico

Recentemente as coálgebras têm vindo a ser consideradas adequadas para descrever a semântica de sistemas dinâmicos; nesse processo tem particular interesse a existência de coálgebras finais. Nesta secção vamos apresentar uma coálgebra final para um functor apropriado que descreve a dinâmica do sistema em estudo – o cálculo-π. Começamos por apresentar o functor base para a construção do domínio com que vamos trabalhar e os *sistemas de passagem de nomes*, que são as coálgebras de conjuntos nominais para esse functor. Em seguida construímos, em vários passos, uma coálgebra final, $\mathbb{D}$, para esse functor, que será o domínio semântico no qual iremos interpretar o cálculo-π.

4.2.1 Sistemas de passagem de nomes

Relembremos aqui o endofunctor, definido na categoria dos conjuntos nominais, introduzido e estudado na secção 2.2:

$$F(Q) = \mathcal{P}_{fin}(Q + (\mathcal{N} \times \mathcal{N} \times Q) + (\mathcal{N} \times Q^{\mathcal{N}}) + (\mathcal{N} \times Q^{\mathcal{N}}))$$

onde $Q^{\mathcal{N}}$ é o conjunto das funções de $\mathcal{N}$ em Q de suporte finito. Note-se que, de acordo com o que vimos atrás na secção 2.2, se Q é um conjunto nominal, $F(Q)$ também o é. Este será o nosso functor principal, à custa do qual se construirá o domínio semântico para o Cálculo-π.

Consideremos agora o endofunctor que constitui a base de F:

$$F_0(Q) = Q + (\mathcal{N} \times \mathcal{N} \times Q) + (\mathcal{N} \times Q^{\mathcal{N}}) + (\mathcal{N} \times Q^{\mathcal{N}})$$

Este functor descreve o tipo das transições do sistema que iremos considerar. A primeira componente corresponde a uma acção interna; a segunda ao envio de um nome por um canal; a terceira ao envio de um nome privado por um canal e a quarta à recepção de um nome por um canal; esta interpretação será tornada mais clara mais adiante. Iremos utilizar a seguinte notação para os elementos de $F_0(Q)$:

$(1, q)$ será denotado por q;
$(2, a, x, q)$ será denotado por $(\overline{a}, x, q)$;
$(3, a, e)$ será denotado por $(\overline{a}, e)$;
$(4, a, t)$ será denotado por (a, t).

Temos que $F(Q) = \mathcal{P}_{fin}(F_0(Q))$.

Definição 4.2.1 (Sistema-pn) Um *sistema de passagem de nomes*, ou simplesmente *sistema-pn*, é um par (Q, ξ) onde Q é um conjunto nominal e $\xi : Q \to F(Q)$, com F o functor acima referido, é um morfismo de conjuntos nominais. □

Os sistema-pn serão usualmente definidos por um sistema de transições etiquetadas, utilizando a seguinte notação:

$$\begin{array}{lll} p \xrightarrow{\tau}_{\xi} q & \Leftrightarrow & q \in \xi(p) \\ p \xrightarrow{\overline{a}}_{\xi} \langle x \rangle q & \Leftrightarrow & (\overline{a}, x, q) \in \xi(p) \\ p \xrightarrow{\overline{a}}_{\xi} e & \Leftrightarrow & (\overline{a}, e) \in \xi(p) \\ p \xrightarrow{a}_{\xi} t & \Leftrightarrow & (a, t) \in \xi(p) \end{array}$$

O símbolo ξ em índice será omitido sempre que for evidente, pelo contexto, qual o sistema envolvido. Uma transição $p \xrightarrow{\tau} q$ corresponde à execução de uma acção interna; $p \xrightarrow{\overline{a}} \langle x \rangle q$ ao envio do nome x pelo canal a; $p \xrightarrow{\overline{a}} e$ a uma continuação e parametrizada por um nome novo a enviar pelo canal a; $p \xrightarrow{a} t$ a uma continuação t que é função do nome a receber pelo canal a.

De acordo com o exposto na subsecção 2.3.2, vamos apresentar a relação de bissimulação para o functor F utilizando a notação que acabamos de introduzir.

Definição 4.2.2 (Relação de bissimulação para sistemas-pn) Sejam P e Q sistemas-pn. Uma relação binária $R \subseteq P \times Q$ é uma *bissimulação* se R é simétrica e pRq implica

1. $p \xrightarrow{\tau} p' \;\Rightarrow\; \exists_{q'}\; q \xrightarrow{\tau} q' \,\&\, p' R\, q'$
2. $p \xrightarrow{\overline{a}} \langle x \rangle p' \;\Rightarrow\; \exists_{q'}\; q \xrightarrow{\overline{a}} \langle x \rangle q' \,\&\, p' R\, q'$
3. $p \xrightarrow{\overline{a}} e \;\Rightarrow\; \exists_{e'}\; q \xrightarrow{\overline{a}} e' \,\&\, \forall_x\; e(x) R\, e'(x)$
4. $p \xrightarrow{a} t \;\Rightarrow\; \exists_{t'}\; q \xrightarrow{a} t' \,\&\, \forall_x\; t(x) R\, t'(x)$

□

4.2.2 Construção do limite

Seja $(\mathcal{U}_n)_{n \geq 0}$ a sucessão de conjuntos nominais

$$\begin{aligned} \mathcal{U}_0 &= \{\emptyset\}; \\ \mathcal{U}_{n+1} &= F(\mathcal{U}_n). \end{aligned}$$

O conjunto $\mathcal{U}_0$ é um conjunto nominal com $\sigma.\emptyset = \emptyset$. $\mathcal{U}_{n+1}$ é um conjunto nominal e uma acção em $\mathcal{U}_{n+1}$ é $\sigma.X = \{\sigma.x : x \in X\}$, para $X \in \mathcal{U}_{n+1} = F(\mathcal{U}_n)$. A aplicação de σ a elementos de $\mathcal{U}_{n+1}$ está bem definida por indução; para $n \in \mathcal{N}$ tem-se $\sigma.n = \sigma(n)$ e para $f : \mathcal{N} \to \mathcal{U}_n$, $(\sigma.f)(n) = \sigma.(f(\sigma^{-1}.n))$.

Conclui-se facilmente por indução que, para todo o n, temos $\mathcal{U}_n \subseteq \mathcal{U}_{n+1}$. Para $n = 0$ temos $\mathcal{U}_0 \subseteq \mathcal{U}_1$ pela definição de F e, assumindo que $\mathcal{U}_n \subseteq \mathcal{U}_{n+1}$, como F é monótono (secção 2.2.6) temos $F(\mathcal{U}_n) \subseteq F(\mathcal{U}_{n+1})$ que é o mesmo que $\mathcal{U}_{n+1} \subseteq \mathcal{U}_{n+2}$.

Seja $(\beta_n)_{n\geq 0}$ a sequência de morfismos de conjuntos-Π definida do seguinte modo

$$\begin{aligned}&\beta_0 : \mathcal{U}_1 \to \mathcal{U}_0 \text{ é o único morfismo possível e}\\ &\beta_{n+1} : \mathcal{U}_{n+2} \to \mathcal{U}_{n+1} \text{ é } F(\beta_n).\end{aligned}$$

Assim,

$$\begin{aligned}\beta_{n+1}(X) = F(\beta_n)(X) = \ & \{\beta_n(q) : q \in X\}\\ & \cup\\ & \{(\overline{a}, x, \beta_n(q)) : (\overline{a}, x, q) \in X\}\\ & \cup\\ & \{(\overline{a}, \beta_n \circ e) : (\overline{a}, e) \in X\}\\ & \cup\\ & \{(a, \beta_n \circ t) : (a, t) \in X\}.\end{aligned}$$

As funções β_n são todas sobrejectivas, pois β_0 é definida sobrejectiva e $\beta_{n+1} = F(\beta_n)$ onde F é um functor, logo preserva a sobrejectividade.

Obtém-se assim uma cadeia

$$\mathcal{U}_0 \xleftarrow{\beta_0} \mathcal{U}_1 \xleftarrow{\beta_1} \mathcal{U}_2 \ldots \mathcal{U}_n \xleftarrow{\beta_n} \mathcal{U}_{n+1} \ldots$$

Define-se $\mathcal{U}_\omega$ como o limite de $(\mathcal{U}_n)_{n\geq 0}$, dado por:

$$\mathcal{U}_\omega = \{(u_n)_{n\geq 0} : \forall_n\ u_n \in \mathcal{U}_n \text{ e } u_n = \beta_n(u_{n+1})\}.$$

Relativamente aos elementos de $\mathcal{U}_\omega$ iremos utilizar as notações $\vec{u} = (u_n)_{n\geq 0}$ e $\vec{u}[n] = u_n$.

Podemos estender a estrutura de conjunto-Π a $\mathcal{U}_\omega$ fazendo $\sigma.\vec{u} = (\sigma.\vec{u}[n])_{n\geq 0}$. Com efeito, tem-se $1.\vec{u} = (1.\vec{u}[n])_{n\geq 0} = (\vec{u}[n])_{n\geq 0} = \vec{u}$ e $\theta.(\sigma.\vec{u}) = \theta.(\sigma.\vec{u}[n])_{n\geq 0} = (\theta.(\sigma.\vec{u}[n]))_{n\geq 0} = ((\theta \circ \sigma).\vec{u}[n])_{n\geq 0} = (\theta \circ \sigma).\vec{u}$.

O conjunto $\mathcal{U}_\omega$ não é um conjunto nominal, embora os $\mathcal{U}_n$ o sejam, uma vez que contém muitos elementos que não são finitamente suportados.

Lema 4.2.3 *Um elemento $\vec{u} \in \mathcal{U}_\omega$ é finitamente suportado se, e só se, a sequência dos suportes $sup(\vec{u}[0]), sup(\vec{u}[1]), ..., sup(\vec{u}[n]), ...$ é constante a partir de determinada ordem, isto é, existe um k tal que, $sup(\vec{u}[k])$ é finito e, para todo o p, $sup(\vec{u}[k+p]) = sup(\vec{u}[k])$. Neste caso tem-se $sup(\vec{u}) = sup(\vec{u}[k])$*

Demonstração Comecemos por verificar que a sequência dos suportes é não decrescente. Ora, $sup(\vec{u}[n]) = sup(\alpha_n(\vec{u}[n+1])) \subseteq sup(\vec{u}[n+1])$ visto que α_n é um morfismo de conjuntos-Π. É fácil verificar que um conjunto $X \subseteq \mathcal{N}$, não necessariamente finito, suporta $\vec{u}$ se, e só se, X suporta cada um dos $\vec{u}[n]$. Logo, a união dos $sup(\vec{u}[n])$ é o menor conjunto que suporta $\vec{u}$. Assim, $sup(\vec{u})$ é finito se, e só se, a sequência dos suportes é constante a partir de determinada ordem e, neste caso, a sequência estabiliza no menor conjunto que suporta $\vec{u}$, isto é, no $sup(\vec{u})$. □

Vamos denotar por $\mathcal{U}$ o conjunto nominal dos elementos de $\mathcal{U}_\omega$ com suporte finito.

Para todo o $n \geq 0$ define-se a função projecção $pr_n : \mathcal{U} \to \mathcal{U}_n$ por $pr_n(\vec{u}) = \vec{u}[n]$. Para cada n, a projecção $pr_n : \mathcal{U} \to \mathcal{U}_n$ é um morfismo de conjuntos-Π uma vez que $pr_n(\vec{u}^\sigma) = \vec{u}^\sigma[n] = \vec{u}[n]^\sigma = pr_n(\vec{u})^\sigma$. É também fácil concluir que $pr_n = \beta_n \circ pr_{n+1}$ para todo o n.
Para n menor que k, define-se $\beta_{kn} : \mathcal{U}_k \to \mathcal{U}_n$ por $\beta_{kn} = \beta_n \circ ... \circ \beta_{k-1}$.

Lema 4.2.4 *Seja $\vec{u} \in \mathcal{U}$ e $k \geq 0$.*

(a) *Seja $v \in \mathcal{U}_k$ com $v \in \vec{u}[k+1]$. Então existe $\vec{v} \in \mathcal{U}$ tal que $\vec{v}[k] = v$ e para todo o n, $\vec{v}[n] \in \vec{u}[n+1]$.*

(b) *Seja $(\overline{a}, x, v) \in \mathcal{N} \times \mathcal{N} \times \mathcal{U}_k$ com $(\overline{a}, x, v) \in \vec{u}[k+1]$. Então existe $\vec{v} \in \mathcal{U}$ tal que $\vec{v}[k] = v$ e para todo o n, $(\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]$.*

(c) *Seja* $(\overline{a}, f) \in \mathcal{N} \times \mathcal{U}_k$ *com* $(\overline{a}, f) \in \vec{u}[k+1]$. *Então existe* $e \in \mathcal{U}^{\mathcal{N}}$ *tal que* $pr_k \circ e = f$ *e para todo o* n, $(\overline{a}, pr_n \circ e) \in \vec{u}[n+1]$.

(d) *Seja* $(a, f) \in \mathcal{N} \times \mathcal{U}_k$ *com* $(a, f) \in \vec{u}[k+1]$. *Então existe* $t \in \mathcal{U}^{\mathcal{N}}$ *tal que* $pr_k \circ t = f$ *e para todo o* n, $(a, pr_n \circ t) \in \vec{u}[n+1]$.

Demonstração

(a) Para $n < k$ fazemos $\vec{v}[n] = \beta_{kn}(v)$, $\vec{v}[k] = v$ e para $n > k$ obtemos os componentes de $\vec{v}$ pelo seguinte processo indutivo: conhecendo $\vec{v}[n]$ elemento de $\mathcal{U}_n$ pertencente a $\vec{u}[n+1]$, pretendemos obter $\vec{v}[n+1]$ elemento de $\mathcal{U}_{n+1}$ pertencente a $\vec{u}[n+2]$, ora como $\vec{u} \in \mathcal{U}$ sabemos que

$$\begin{aligned}\vec{u}[n+1] = \beta_{n+1}(\vec{u}[n+2]) = \; & \{\beta_n(q) : q \in \vec{u}[n+2]\} \\ & \cup \\ & \{(a, x, \alpha_n(q)) : (\overline{a}, x, q) \in \vec{u}[n+2]\} \\ & \cup \\ & \{(\overline{a}, \beta_n \circ e) : (\overline{a}, e) \in \vec{u}[n+2]\} \\ & \cup \\ & \{(a, \beta_n \circ t) : (a, t) \in \vec{u}[n+2]\}.\end{aligned}$$

Como $\vec{v}_n \in \mathcal{U}_n$ ele pertence ao primeiro conjunto, logo existe um $q \in \vec{u}[n+2]$ tal que $\beta_n(q) = \vec{v}[n]$ e tomo $\vec{v}[n+1] = q$ que é um elemento de $\vec{u}[n+2]$ que está em $\mathcal{U}_{n+1}$. O $\vec{v} = (\vec{v}_n)_{n \geq 0}$ obtido desta forma verifica a propriedade pretendida e é um elemento de $\mathcal{U}$ pois a própria construção garante que para todo o n, $\vec{v}_n \in \mathcal{U}_n$ e $\beta_n(\vec{v}_{n+1}) = \vec{v}_n$.

(b) Construa-se $\vec{v}$ da seguinte forma:

$\vec{v}[k] = v$;

para $n < k$, $\vec{v}[n] = \beta_{kn}(v)$;

para $n > k$, obtemos $\vec{v}[n]$ por um processo indutivo. Como $\beta_n(\vec{u}[n+1]) = \vec{u}[n]$ sabemos que existe $Y \in \vec{u}[n+1]$ tal que $\beta_n(Y) = (\overline{a}, x, \vec{v}[n-1])$. Atendendo à definição de β_n a única possibilidade é ser $Y = (\overline{a}, x, q)$ com $\vec{v}[n-1] = \beta_{n-1}(q)$ e defina-se $\vec{v}[n] = q$.

Esta construção garante-nos que $\forall_n\,(\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]$.

(c) Construa-se e da seguinte forma:

$pr_k \circ e = f$;

para $n < k$, $pr_n \circ e = \beta_{kn} \circ f$;

para $n > k$, obtemos $pr_n \circ e$ por um processo indutivo. Como $\beta_n(\vec{u}[n+1]) = \vec{u}[n]$ sabemos que existe $Y \in \vec{u}[n+1]$ tal que $\beta_n(Y) = (\overline{a}, pr_{n-1} \circ e)$. Atendendo à definição de β_n a única possibilidade é ser $Y = (\overline{a}, g)$ com $pr_{n-1} \circ e = \beta_{n-1} \circ g$ e defina-se $pr_n \circ e = g$.

Esta construção garante-nos que $\forall_n\,(\overline{a}, pr_n \circ e) \in \vec{u}[n+1]$.

(d) É análoga à anterior. □

Seja G o endofunctor sobre conjuntos nominais dado por

$$G(Q) = \mathcal{P}_{sf}(Q + (\mathcal{N} \times \mathcal{N} \times Q) + (\mathcal{N} \times Q^{\mathcal{N}}) + (\mathcal{N} \times Q^{\mathcal{N}}))$$

onde se recorda que $\mathcal{P}_{sf}$ é o functor das potências de suporte finito. Ora G é igual a F à excepção de usar as partes com suporte finito do conjunto onde F usa as partes finitas do conjunto e logo $F(Q) \subseteq G(Q)$.

Definição 4.2.5 (Função χ) Seja

$$\chi : \mathcal{U} \to G(\mathcal{U})$$

a função definida por

$$\begin{aligned}\chi(\vec{u}) \;=\; & \{\vec{v} : \forall_n \vec{v}[n] \in \vec{u}[n+1]\} \\ & \cup \\ & \{(\overline{a}, x, \vec{v}) : \forall_n (\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]\} \\ & \cup \\ & \{(\overline{a}, e) : \forall_n (\overline{a}, pr_n \circ e) \in \vec{u}[n+1]\} \\ & \cup \\ & \{(a, t) : \forall_n (a, pr_n \circ t) \in \vec{u}[n+1]\}.\end{aligned}$$

□

Atendendo ao isomorfismo existente entre $\mathcal{P}_{sf}(X+Y)$ e $\mathcal{P}_{sf}(X) \times \mathcal{P}_{sf}(Y)$, podemos escrever χ como

$$\chi = \chi_\tau \cup \chi_{fo} \cup \chi_{bo} \cup \chi_{in}$$

onde

$$\begin{aligned}\chi_\tau : \;\; \mathcal{U} &\to \mathcal{P}_{sf}(\mathcal{U}) \\ \vec{u} &\mapsto \{\vec{v} : \forall_n \vec{v}[n] \in \vec{u}[n+1]\}\end{aligned}$$

$$\begin{aligned}\chi_{fo} : \;\; \mathcal{U} &\to \mathcal{P}_{sf}(\mathcal{N} \times \mathcal{N} \times \mathcal{U}) \\ \vec{u} &\mapsto \{(\overline{a}, x, \vec{v}) : \forall_n (\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]\}\end{aligned}$$

$$\begin{aligned}\chi_{bo} : \;\; \mathcal{U} &\to \mathcal{P}_{sf}(\mathcal{N} \times \mathcal{P}_{fin}(\mathcal{N}) \times \mathcal{U}^{\mathcal{N}}) \\ \vec{u} &\mapsto \{(\overline{a}, e) : \forall_n (\overline{a}, pr_n \circ e) \in \vec{u}[n+1]\}\end{aligned}$$

$$\begin{aligned}\chi_{in} : \;\; \mathcal{U} &\to \mathcal{P}_{sf}(\mathcal{N} \times \mathcal{U}^{\mathcal{N}}) \\ \vec{u} &\mapsto \{(a, t) : \forall_n (a, pr_n \circ t) \in \vec{u}[n+1]\}\end{aligned}$$

Lema 4.2.6 *A função χ está bem definida, isto é,*

$$\forall_{\vec{u} \in \mathcal{U}},\ \chi(\vec{u}) \text{ é finitamente suportado.}$$

Mais precisamente, $\chi(\vec{u})$ é suportado por $sup(\vec{u})$.

Demonstração Para garantir o resultado basta-nos provar que todo o elemento de $\chi(\vec{u})$ é suportado por $sup(\vec{u})$. Vamos ver em pormenor os casos em que os elementos de $\chi(\vec{u})$ são da forma $\vec{v}$ e $(\overline{a}, e)$, os restantes dois casos demonstram-se de modo idêntico.

No caso caso $\vec{v} \in \chi(\vec{u})$ tem-se, para todo o n, $\vec{v}[n] \in \vec{u}[n+1]$. Como $\vec{u}[n+1]$ é um conjunto finito o seu suporte é a união dos suportes dos seus elementos. Isto implica que, para todo o n, $sup(\vec{v}[n]) \subseteq sup(\vec{u}[n+1]) \subseteq sup(\vec{u})$ e logo $sup(\vec{v}) \subseteq sup(\vec{u})$.

No caso de $(\overline{a}, e) \in \chi(\vec{u})$ tem-se que, para todo o n, $(\overline{a}, pr_n \circ e) \in \vec{u}[n+1]$, logo $sup((\overline{a}, pr_n \circ e)) \subseteq sup(\vec{u}[n+1]) \subseteq sup(\vec{u})$, donde $a \in sup(\vec{u})$ e $sup(e) \subseteq sup(\vec{u})$ e portanto $sup((\overline{a}, e)) \subseteq sup(\vec{u})$ como se pretendia. □

Lema 4.2.7 *A função $\chi : \mathcal{U} \to G(\mathcal{U})$ é um morfismo de conjuntos-*Π.

Demonstração Temos de mostrar que $\chi(\vec{u^\sigma}) = \chi(\vec{u})^\sigma$ para todo o $\vec{u} \in \mathcal{U}$ e $\sigma \in \Pi$. Consideremos $\chi(\vec{u^\sigma}) = \chi_\tau(\vec{u^\sigma}) \cup \chi_{fo}(\vec{u^\sigma}) \cup \chi_{bo}(\vec{u^\sigma}) \cup \chi_{in}(\vec{u^\sigma})$, o resultado fica assegurado se demonstrarmos que cada uma das funções que compõem χ é um morfismo de conjuntos-Π. Comecemos por verificar a afirmação para χ_τ:

$$\begin{aligned}
\chi_\tau(\vec{u^\sigma}) &= \{\vec{v} : \forall_n\ \vec{v}[n] \in \vec{u^\sigma}[n+1]\} \\
&= \{\vec{v} : \forall_n\ \vec{v^{\sigma^{-1}}}[n] \in \vec{u}[n+1]\} \\
&= \{\vec{w^\sigma} : \forall_n\ \vec{w}[n] \in \vec{u^\sigma}[n+1]\} \\
&= \chi_\tau(\vec{u})^\sigma
\end{aligned}$$

Para χ_{bo}, como $(pr_n \circ e)^{\sigma^{-1}} = pr_n \circ e^{\sigma^{-1}}$ pelo facto de $.^{\mathcal{N}}$ ser um functor,

tem-se:

$$\begin{aligned}\chi_{bo}(\vec{u}^{\sigma}) &= \{(\overline{a}, e) : \forall_n\, (\overline{a}, pr_n \circ e) \in \vec{u}^{\sigma}[n+1]\} \\ &= \{(\overline{a}, e) : \forall_n\, (\overline{a}^{\,\sigma^{-1}}, pr_n \circ e^{\,\sigma^{-1}}) \in \vec{u}[n+1]\} \\ &= \{(\overline{a}^{\,\sigma}, e^{\sigma}) : \forall_n\, (\overline{a}, pr_n \circ e) \in \vec{u}[n+1]\} \\ &= \chi_{bo}(\vec{u})^{\sigma}.\end{aligned}$$

Os restantes dois casos demonstram-se de modo semelhante. □

Vamos agora apresentar um resultado auxiliar que será uma referência útil para várias demonstrações.

Lema 4.2.8 $pr_{k+1} = F(pr_k) \circ \chi$.

Demonstração Sabemos que

$$\begin{aligned}F(pr_k)(X) = \; & \{\vec{v}[k] : \vec{v} \in X\} \\ & \cup \\ & \{(\overline{a}, x, \vec{v}[k]) : (\overline{a}, x, v) \in X\} \\ & \cup \\ & \{(\overline{a}, pr_k \circ e) : (\overline{a}, e) \in X\} \\ & \cup \\ & \{(a, pr_k \circ t) : (\overline{a}, t) \in X\}.\end{aligned}$$

Aplicando $F(pr_k)$ a ambos os membros da definição de χ obtemos, por

definição de $F(pr_k)$,

$$\begin{aligned} F(pr_k)(\chi(\vec{u})) = & \{\vec{v}[k] : \forall_n \vec{v}[n] \in \vec{u}[n+1]\} \\ & \cup \\ & \{(\overline{a}, x, \vec{v}[k]) : \forall_n (\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]\} \\ & \cup \\ & \{(\overline{a}, pr_k \circ e) : \forall_n (\overline{a}, pr_n \circ e) \in \vec{u}[n+1]\} \\ & \cup \\ & \{(a, pr_k \circ t) : \forall_n (a, pr_n \circ t) \in \vec{u}[n+1]\} \\ = & \ \vec{u}[k+1] \\ = & \ pr_{k+1}(\vec{u}). \end{aligned}$$

A justificação para a segunda igualdade é a seguinte:
A inclusão $\subseteq$ é imediata pois basta considerarmos $n = k$ na condição que define cada um dos conjuntos, temos então de mostrar que

$$\begin{aligned} \vec{u}[k+1] \subseteq & \{\vec{v}[k] : \forall_n \vec{v}[n] \in \vec{u}[n+1]\} \\ & \cup \\ & \{(\overline{a}, x, \vec{v}[k]) : \forall_n (\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]\} \\ & \cup \\ & \{(\overline{a}, pr_k \circ e) : \forall_n (\overline{a}, pr_n \circ e) \in \vec{u}[n+1]\} \\ & \cup \\ & \{(a, pr_k \circ t) : \forall_n (a, pr_n \circ t) \in \vec{u}[n+1]\} \end{aligned}$$

Seja $X \in \vec{u}[k+1]$ temos a considerar quatro casos. Se $X \in \mathcal{U}_k$ estamos nas condições do lema 4.2.4(a) e portanto existe $\vec{v} \in \mathcal{U}$ tal que $\vec{v}[k] = X$ e $\forall_n \vec{v}[n] \in \vec{u}[n+1]$ o que garante que X está no primeiro conjunto. Se $X \in \mathcal{N} \times \mathcal{N} \times \mathcal{U}_k$ estamos nas condições do lema 4.2.4(b) e portanto existe $\vec{v} \in \mathcal{U}$ tal que $\vec{v}[k] = X$ e $\forall_n (\overline{a}, x, \vec{v}[n]) \in \vec{u}[n+1]$ o que garante que X está no segundo conjunto. Do mesmo modo, se $X \in \mathcal{N} \times \mathcal{U}_k^{\mathcal{N}}$ estamos nas condições do lema 4.2.4(c) ou (d), o que garante que X está no terceiro ou quarto conjunto, respectivamente. □

4.2.3 O sistema-pn final

Nesta subsecção vamos extrair de $\mathcal{U}$ os elementos finitamente ramificados, verificar que ainda obtemos um conjunto-Π e que o conjunto dos elementos finitamente ramificados, com a correspondente restrição de χ, constitui um sistema de passagem de nomes final.

Definição 4.2.9 (Finitamente ramificado) Seja X um subconjunto-Π de $\mathcal{U}$. Diz-se que X é *finitamente ramificado* se $\chi(\vec{u}) \in F(X)$ para todo o $\vec{u} \in X$, isto é, χ restringe-se a uma função de X em $F(X)$. □

O lema que se segue garante que o conjunto obtido extraindo de $\mathcal{U}$ apenas os elementos finitamente ramificados ainda é um conjunto-Π.

Lema 4.2.10 *Se $(X_i)_{i\in I}$ é uma família (possivelmente vazia) de subconjuntos-Π de $\mathcal{U}$ finitamente ramificados, a sua união $\bigcup_i X_i$ é ainda um subconjunto-Π de $\mathcal{U}$ finitamente ramificado.*

Demonstração O resultado é imediato atendendo a que neste caso χ restringe-se a uma função de $\bigcup_i X_i$ em $F(\bigcup_i X_i)$. □

Definição 4.2.11 (Domínio $\mathbb{D}$) O maior subconjunto-Π de $\mathcal{U}$ finitamente ramificado é denotado por $\mathbb{D}$. A restrição de χ a uma função de $\mathbb{D}$ em $F(\mathbb{D})$ será ainda designada por χ. □

Lema 4.2.12 *$\mathbb{D}$ é um conjunto-Π e χ é um morfismo de conjuntos-Π.*

Demonstração Pela própria definição de $\mathbb{D}$. □

Vamos agora mostrar que $(\mathbb{D}, \chi)$ é uma coálgebra final para o functor que descreve as transições dos termos do cálculo-π.

Proposição 4.2.13 $(\mathbb{D}, \chi)$ *é um sistema-pn final.*

Demonstração Temos de mostrar que para todo o sistema-pn (Q, ϕ) existe um único morfismo $f : (Q, \phi) \to (\mathbb{D}, \chi)$. Para todo o $q \in Q$ tomamos para $f(q)$ o elemento de $\mathbb{D}$ definido por

$$\begin{aligned} f(q)[0] &= \emptyset, \\ f(q)[n+1] &= \{f(p)[n] : p \in \phi_\tau(q)\} \\ &\quad \cup \\ &\quad \{(\overline{a}, x, f(p)[n]) : (\overline{a}, x, p) \in \phi_{fo}(q)\} \\ &\quad \cup \\ &\quad \{(\overline{a}, pr_n \circ f \circ e) : (\overline{a}, e) \in \phi_{bo}(q)\} \\ &\quad \cup \\ &\quad \{(a, pr_n \circ f \circ t) : (a, t) \in \phi_{in}(q)\}. \end{aligned}$$

Temos de verificar que f está bem definida e é o morfismo único que procuramos. Em termos de notação, iremos escrever $\lambda_x f(e(x))[n])$ e $\lambda_x f(t(x))[n])$ em vez de $pr_n \circ f \circ e$ e $pr_n \circ f \circ t$, respectivamente.
Vamos organizar a prova em vários pontos: (i) f está bem definida; (ii) f é um morfismo de sistemas-pn; (iii) f é único.

(i) f está bem definida

1. Vamos começar por verificar que $f(q) \in \mathcal{U}$ para todo o $q \in Q$. Para isto temos de provar que:

   (a) $f(q)[n] \in \mathcal{U}_n$ para todo o n:

   Temos $f(q)[0] = \emptyset \in \mathcal{U}_0$. Assumindo o resultado para n, vamos mostrar que cada um dos quatro conjuntos que define $f(q)[n+1]$ está em $F(\mathcal{U}_n)$. Os conjuntos são todos finitos porque $\phi(q)$ é um conjunto finito.

   Para o primeiro dos quatro conjuntos, por hipótese de indução

tem-se que $f(p)[n]$ está em $\mathcal{U}_n$, logo $\{f(p)[n] : p \in \phi_\tau(q)\}$ é um subconjunto finito de $\mathcal{U}_n$ e portanto está em $F(\mathcal{U}_n)$.
Para o último dos conjuntos, por hipótese de indução, a função $\lambda_x f(t(x))[n]$ aplica $\mathcal{N}$ em $\mathcal{U}_n$, logo temos um subconjunto finito de $\mathcal{N} \times \mathcal{U}^{\mathcal{N}}$ e portanto um elemento de $F(\mathcal{U}_n)$.
Os outros casos demonstram-se de modo similar.

(b) $\beta_n(f(q)[n+1]) = f(q)[n]$ para todo o n:

Para $n = 0$ o resultado é imediato. Vamos assumir, como passo de indução, que $\beta_{n+1} = F(\beta_n)$ e temos

$$\begin{aligned}\beta_{n+1}(f(q)[n+2]) = \; & \{\beta_n(f(p)[n+1]) : p \in \phi_\tau(q)\} \\ & \cup \\ & \{(\overline{a}, x, \beta_n(f(p)[n+1])) : (\overline{a}, x, p) \in \phi_{fo}(q)\} \\ & \cup \\ & \{(\overline{a}, \beta_n \circ \lambda_x f(e(x))[n+1]) : (\overline{a}, e) \in \phi_{bo}(q)\} \\ & \cup \\ & \{(a, \beta_n \circ \lambda_x f(t(x))[n+1]) : (a, t) \in \phi_{in}(q)\}.\end{aligned}$$

Por indução, $\beta_n(f(p)[n+1]) = f(p)[n]$ e $\beta_n(f(e(x))[n+1]) = f(e(x))[n]$, pelo que $\beta_n \circ \lambda_x f(e(x))[n+1] = \lambda_x f(e(x))[n]$ donde se obtém o resultado pretendido.

2. $f(Q)$ é um subconjunto-Π de $\mathcal{U}$ finitamente ramificado e portanto $f(Q) \subseteq \mathbb{D}$:

Temos de verificar que para todo o $q \in Q$, $\chi(f(q))$ está em $F(f(Q))$.

Para o efeito basta-nos mostrar que

$$\begin{aligned}\chi(f(q)) = \ & \{f(p) : p \in \phi_\tau(q)\} \\ & \cup \\ & \{(\overline{a}, x, f(p)) : (\overline{a}, x, p) \in \phi_{fo}(q)\} \\ & \cup \\ & \{(\overline{a}, f \circ e) : (\overline{a}, e) \in \phi_{bo}(q)\} \\ & \cup \\ & \{(a, f \circ t) : (a, t) \in \phi_{in}(q)\}.\end{aligned}$$

De facto, como $f(p) \in f(Q)$; $f \circ e, f \circ t \in f(Q)^{\mathcal{N}}$ e cada um dos quatro conjuntos da direita é finito porque $\phi(q)$ é finito, tem-se que $\chi(f(q))$ está em $F(f(Q))$.

Para provar a igualdade apresentada vamos atender a que $\chi(f(q))$ é a união de quatro conjuntos, e cada um deles terá de ser igual ao correspondente da igualdade. Vamos apresentar a demonstração apenas para o caso χ_{in} (o quarto conjunto) uma vez que os outros casos são semelhantes. Temos então de verificar que

$$\{(a, t) : \forall_n \ (a, pr_n \circ t) \in f(q)[n+1]\} = \{(a, f \circ t) : (a, t) \in \phi_{in}(q)\}.$$

Vamos começar por verificar a inclusão $\supseteq$. Assim, dado $(a, t) \in \phi_{in}(q)$ temos de verificar que $(a, f \circ t)$ está no conjunto da esquerda, isto é, para todo o n, $(a, pr_n \circ f \circ t) \in f(q)[n+1]$. Por definição de pr_n podemos reescrever a condição como $(a, \lambda_x f(t(x)))[n] \in f(q)[n+1]$, e agora, por definição de $f(q)[n+1]$, é imediata a sua veracidade.

Para provar a inclusão $\subseteq$ tomamos (a, t) no conjunto da esquerda e mostramos que existe t' tal que $(a, t') \in \phi_{in}(q)$ e $t = f \circ t'$. Ora se (a, t) está no conjunto da esquerda tem-se $(a, pr_n \circ t) \in f(q)[n+1]$ para todo o n. Por definição de $f(q)[n+1]$, existe t'_n tal que $(a, t'_n) \in \phi_{in}(q)$ e $pr_n \circ t = pr_n \circ f \circ t'_n$. Como $\phi_{in}(q)$ é finito e contém os

pares $(a, t'_0), \ldots, (a, t'_n), \ldots$, pelo menos um dos elementos $t'_0, \ldots, t'_n, \ldots$ repete-se infinitas vezes, vamos designar este elemento por t'. Isto implica que se verifica igualdade $pr_n \circ t = pr_n \circ f \circ t'$ para um número infinito de n's e portanto a igualdade verifica-se para todo o n porque se ela se verifica para $n+1$, então também se verifica para n porque $pr_{n+1} \circ t = pr_{n+1} \circ f \circ t'$ implica $\beta_n \circ pr_{n+1} \circ t = \beta_n \circ pr_{n+1} \circ f \circ t'$ e logo $pr_n \circ t = pr_n \circ f \circ t'$. Assim, para todo x e n temos $t(x)[n] = f(t'(x))[n]$ e portanto $t(x) = f(t'(x))$, logo $t = f \circ t'$, como pretendíamos.

3. f é um morfismo de conjuntos-Π:

   Para todo $q \in Q$ e $\sigma \in \Pi$, temos de verificar que $f(q^\sigma) = f(q)^\sigma$, isto é, $f(q^\sigma)[n] = f(q)^\sigma[n] = f(q)[n]^\sigma$ para todo o n. Para $n = 0$ tem-se $f(q^\sigma)[0] = \emptyset = \emptyset^\sigma = f(q)[0]^\sigma$. Assumindo a afirmação para n, como $f(q^\sigma)[n+1]$ e $f(q)[n+1]^\sigma$ são a união de quatro conjuntos, vamos provar que os conjuntos correspondentes são iguais. Vamos ver em pormenor apenas o caso do último conjunto uma vez que os restantes casos se provam de modo semelhante:

$$\begin{aligned}\{(a, pr_n \circ f \circ t) : (a,t) \in \phi_{in}(q^\sigma)\} &= \{(a, pr_n \circ f \circ t) : (a,t) \in \phi_{in}(q)^\sigma\} \\ &= \{(a^\sigma, pr_n \circ f \circ t^\sigma) : (a,t) \in \phi_{in}(q)\} \\ &= \{(a^\sigma, (pr_n \circ f \circ t)^\sigma) : (a,t) \in \phi_{in}(q)\} \\ &= \{(a, pr_n \circ f \circ t) : (a,t) \in \phi_{in}(q)\}^\sigma.\end{aligned}$$

(ii) **f é um morfismo de sistemas-pn**

É necessário verificar que $\chi \circ f = F(f) \circ \phi$, isto é, que $\chi(f(q)) = F(f)(\phi(q))$ para todo o $q \in Q$. Pela definição de $F(f)$, isto já foi provado quando mostramos que $f(Q)$ é finitamente ramificado.

(iii) f **é único**

Suponhamos que $g : Q \to \mathbb{D}$ é outro morfismo tal que $\chi \circ g = F(g) \circ \phi$. Atendendo à igualdade $pr_{n+1} = F(pr_n) \circ \chi$ temos:

$$\begin{aligned} pr_{n+1} \circ g &= F(pr_n) \circ \chi \circ g \\ &= F(pr_n) \circ F(g) \circ \phi \\ &= F(pr_n \circ g) \circ \phi. \end{aligned}$$

Do mesmo modo se concluia que $pr_{n+1} \circ f = F(pr_n \circ f) \circ \phi$. Podemos agora mostrar por indução que $pr_n \circ g = pr_n \circ f$ para todo o n, e portanto $g = f$. Para $n = 0$ tem-se $pr_0 \circ g = \emptyset = pr_n \circ f$. Assumindo a afirmação verdadeira para n, vem $pr_{n+1} \circ g = F(pr_n \circ g) \circ \phi = F(pr_n \circ f) \circ \phi = pr_{n+1} \circ f$, como pretendíamos.

Com isto completamos a demonstração. □

4.3 Semântica operacional

Nesta secção vamos apresentar uma semântica operacional para o cálculo-π, que será definida por uma coálgebra para o functor F. Vamos começar por verificar que o conjunto dos termos do cálculo-π, Proc , constitui um conjunto nominal. Em seguida apresentamos o sistema de transições etiquetadas associado a Proc, que o tornam um sistema-pn, e definimos a semântica operacional do cálculo-π como o único morfismo definido por esse sistema de transições na coálgebra final $(\mathbb{D}, \chi)$.

Conforme se constata na definição que se segue, a aplicação de uma acção de permutação a expressões de processo do cálculo-π, consiste na troca de nomes nas expressões de processo.

Definição 4.3.1 (Aplicação de permutações a processos) Dada $\sigma \in \Pi$

e $P \in \mathsf{Proc}$, define-se $\sigma.P$ indutivamente por:

$$\begin{array}{rr} \sigma.\mathsf{0} = \mathsf{0} & \sigma.(P+P) = \sigma.P + \sigma.Q \\ \sigma.(\tau.P) = \tau.\sigma.P & \sigma.(P|P) = \sigma.P|\sigma.Q \\ \sigma.(a(x).P) = a^{\sigma}(x^{\sigma}).\sigma.P & \sigma.(\nu x)P = (\nu x^{\sigma})\sigma.P \\ \sigma.(\overline{a}\langle x\rangle.P) = \overline{a^{\sigma}}\langle x^{\sigma}\rangle.\sigma.P & \sigma.(!P) = !\sigma.P \end{array}$$

□

Lema 4.3.2 *O conjunto* Proc*, com a acção acima definida, é um conjunto-*Π*.*

Demonstração É imediato que $1.P = P$. Quanto à segunda condição, a demonstração faz-se por indução na estrutura do processo, atendendo a que o conjunto dos nomes, $\mathcal{N}$, é um conjunto- -Π . Assim, para o processo nulo $\theta.(\sigma.\mathsf{0}) = \theta.\mathsf{0} = \mathsf{0} = (\theta \circ \sigma).\mathsf{0}$. Para o prefixo τ vem, $\theta.(\sigma.(\tau.P)) = \theta.(\tau.P^{\sigma}) = \tau.(P^{\sigma})^{\theta} = \tau.P^{\theta\circ\sigma} = (\theta \circ \sigma).(\tau.P)$, como pretendíamos. Para o prefixo $\overline{x}\langle y\rangle$ tem-se $\theta.(\sigma.(\overline{x}\langle y\rangle.P)) = \theta.(\overline{x^{\sigma}}\langle y^{\sigma}\rangle.P^{\sigma}) = \overline{(x^{\sigma})^{\theta}}\langle (y^{\sigma})^{\theta}\rangle.(P^{\sigma})^{\theta} = \overline{x^{\theta\circ\sigma}}\langle y^{\theta\circ\sigma}\rangle.P^{\theta\circ\sigma} = (\theta \circ \sigma).(\overline{x}\langle y\rangle.P)$. O caso do prefixo $x(y)$ é análogo. Para a soma e a composição tomamos $_{op} \in \{+, |\}$ e tem-se $\theta.(\sigma.(P \,_{op}\, Q) = (P^{\sigma})^{\theta} \,_{op}\, (Q^{\sigma})^{\theta}$ que por indução na estrutura de P e Q é o mesmo que $P^{\theta\circ\sigma} \,_{op}\, Q^{\theta\circ\sigma}$ que por sua vez é $(\theta\circ\sigma).(P \,_{op}\, Q)$. Para a restrição, $\theta.(\sigma.(\nu x)P) = (\nu(x^{\sigma})^{\theta})(P^{\sigma})^{\theta}$ que, atendendo a que $x \in \mathcal{N}$, que é um conjunto-Π, e à indução na estrutura de P, é $(\nu x^{\theta\circ\sigma})P^{\theta\circ\sigma} = (\theta \circ \sigma).(\nu x)P$. Finalmente, e também por indução na estrutura de P, tem-se $\theta.(\sigma.!P) = !(P^{\sigma})^{\theta} = !P^{\theta\circ\sigma} = (\theta \circ \sigma).!P$. □

Antes de passarmos à definição do sistema de transições etiquetadas que dá origem à semântica operacional temos de introduzir algumas definições de congruências.

Definição 4.3.3 (Relação de Congruência) Uma relação binária $\cong$ definida em Proc é uma congruência se for uma relação de equivalência e para

todo o $P,\ Q,\ R,\ S \in \mathsf{Proc}$,

$$
\begin{array}{l}
P \cong Q \Rightarrow \pi.P \cong \pi.Q \\
P \cong Q,\ R \cong S \Rightarrow P \mathrel{op} R \cong Q \mathrel{op} S, \quad op \in \{+, |\} \\
P \cong Q \Rightarrow (\nu x)P \cong (\nu x)Q \\
P \cong Q \Rightarrow !P \cong !Q
\end{array}
$$

□

De notar que a congruência-α é uma relação de congruência.

Definição 4.3.4 (Congruência estrutural) Congruência estrutural, denotada por $\equiv$ é a menor relação de congruência entre processos que satisfaz:

$$
\begin{array}{ll}
(estrut\ alfa) & P \equiv_\alpha Q \Rightarrow P \equiv Q \\
(estrut\ som\ nulo) & P + \mathsf{0} \equiv P \\
(estrut\ som\ comu) & P + Q \equiv Q + P \\
(estrut\ som\ assoc) & P + (Q + R) \equiv (P + Q) + R \\
(estrut\ par\ nulo) & P|\mathsf{0} \equiv P \\
(estrut\ par\ comu) & P|Q \equiv Q|P \\
(estrut\ par\ assoc) & P|(Q|R) \equiv (P|Q)|R \\
(estrut\ res\ par) & x \notin lv(P) \Rightarrow P|(\nu x)Q \equiv (\nu x)(P|Q) \\
(estrut\ repl\ copi) & !P \equiv !P|P
\end{array}
$$

□

Note-se que se $P \equiv Q$ então $lv(P) = lv(Q)$, [Mil99]. Na definição da congruência estrutural é usual considerar também as condições $(\nu x)0 \equiv 0$ e $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$. Não o fizemos, porém, porque elas não são imediatamente satisfeitas pelo nosso modelo: requereriam uma passagem ao cociente, o que complicaria desnecessariamente o modelo. Vamos também apresentar a relação de congruência para o functor de base definido em 4.2.2 e que descreve o tipo das transições do sistema.

Definição 4.3.5 (Congruência em $F_0(\mathbf{Proc})$) Sejam $\delta, \delta' \in F_0(\mathsf{Proc})$. Tem-se δ congruente com δ', e denota-se por $\delta \equiv \delta'$, se:

$$
\begin{array}{ll}
(cong\ \tau) & \delta = P \Rightarrow \delta' = Q \,\&\, P \equiv Q; \\
(cong\ out) & \delta = (\overline{a}, x, P) \Rightarrow \delta' = (\overline{a}, x, Q) \,\&\, P \equiv Q; \\
(cong\ bo) & \delta = (\overline{a}, \lambda_z P\{z/w\}) \Rightarrow \delta' = (\overline{a}, \lambda_z Q\{z/w\}), \\
 & \qquad\qquad \forall_x\ P\{x/w\} \equiv Q\{x/w\}; \\
(cong\ in) & \delta = (a, \lambda_z P\{z/w\}) \Rightarrow \delta' = (a, \lambda_z Q\{z/w\}), \\
 & \qquad\qquad \forall_x\ P\{x/w\} \equiv Q\{x/w\}.
\end{array}
$$

□

Vamos definir a semântica operacional do cálculo-π com base num sistema de transições etiquetadas, $P \xrightarrow{\alpha} \delta$, onde δ é um elemento de $F_0(\mathsf{Proc})$ e a etiqueta α pode tomar um valor em $\{a, \overline{a}, \tau\}$, onde a é um nome de $\mathcal{N}$. O caso $\alpha = \tau$ corresponde a uma reacção interna, os casos $\alpha = a$ e $\alpha = \overline{a}$ correspondem à capacidade de P participar numa reacção com um processo concorrente que execute uma transição complementar.

Definição 4.3.6 (Sistema de transições em Proc) Consideremos o sistema de transições etiquetadas, $\gamma : Proc \to F(Proc)$, cuja especificação é definida pelas seguintes regras e axiomas:

Congruência:

$$(\text{cong})\ \frac{P \equiv P' \qquad P' \xrightarrow{\alpha} \delta' \qquad \delta' \equiv \delta}{P \xrightarrow{\alpha} \delta}$$

Prefixo:

$$(\text{in})\ \frac{}{a(x).P \xrightarrow{a} \lambda_z P\{z/x\}}$$

$$(\text{out})\ \frac{}{\overline{a}\langle x\rangle.P \xrightarrow{\overline{a}} < x > P}$$

(tau) $\dfrac{}{\tau.P \xrightarrow{\tau} P}$

Restrição:

$(\nu_\tau)\ \dfrac{P \xrightarrow{\tau} P'}{(\nu x)P \xrightarrow{\tau} (\nu x)P'}$

$(\nu_{out})\ \dfrac{P \xrightarrow{\overline{a}} < y > P'}{(\nu x)P \xrightarrow{\overline{a}} < y > (\nu x)P'}\ \ x \neq a, x \neq y$

$(\nu_{bo})\ \dfrac{P \xrightarrow{\overline{a}} < x > P'}{(\nu x)P \xrightarrow{\overline{a}} \lambda_z P'\{z/x\}}\ \ x \neq a$

$(\nu_{bo\,in})\ \dfrac{P \xrightarrow{\hat{a}} \lambda_z Q\{z/w\}}{(\nu x)P \xrightarrow{\hat{a}} \lambda_z((\nu x)Q)\{z/w\}}\ \ \hat{a} \in \{a, \overline{a}\},\ x \neq a, x \neq w$

Soma:

(sum) $\dfrac{P_i \xrightarrow{\alpha} \delta}{P_1 + P_2 \xrightarrow{\alpha} \delta}\ \ i = 1, 2,\ \ \alpha \in \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$

Composição:

$(\text{comp}_\tau)\ \dfrac{P \xrightarrow{\tau} P'}{P|Q \xrightarrow{\tau} P'|Q}$

$(\text{comp}_{out})\ \dfrac{P \xrightarrow{\overline{a}} < x > P'}{P|Q \xrightarrow{\overline{a}} < x > (P'|Q)}$

$(\text{comp}_{bo\,in})\ \dfrac{P \xrightarrow{\hat{a}} \lambda_z P'\{z/w\}}{P|Q \xrightarrow{\hat{a}} \lambda_z(P'|Q)\{z/w\}},\quad \hat{a} \in \{a, \overline{a}\},\ \ w \notin lv(Q)$

Comunicação:

(sync) $\dfrac{P \xrightarrow{\overline{a}} < x > P' \quad Q \xrightarrow{a} \lambda_z Q'\{z/w\}}{P|Q \xrightarrow{\tau} P'|Q'\{x/w\}}$

$$(\text{sync}_\nu)\ \frac{P \xrightarrow{\overline{a}} \lambda_z P'\{z/w\} \quad Q \xrightarrow{a} \lambda_z Q'\{z/w\}}{P|Q \xrightarrow{\tau} (\nu x)(P'\{x/w\}|Q'\{x/w\})}\ \ x \notin (lv(P') \cup lv(Q')) - \{w\}$$

□

As regras estão divididas em seis grupos. O primeiro grupo tem apenas uma regra que garante que se um processo tem uma transição com etiqueta α para uma expressão δ', então qualquer processo que lhe seja congruente também tem uma transição com α para uma expressão δ congruente com δ'. No segundo grupo estão os axiomas que envolvem os prefixos, que constituem as regras base aonde todas as outras acabam inevitavelmente por conduzir. No caso do prefixo de entrada, é recebido pelo canal a um nome e o sistema prossegue como uma função que tem como argumento o nome que irá ser recebido aquando da interacção do processo com outro a correr concorrentemente. Para o prefixo de saída, é transmitido o nome x pelo canal a (denota-se $\overline{a}$) e o processo fica a aguardar uma acção complementar por parte de outro processo. Quando temos um prefixo de acção interna há uma transição com etiqueta τ e o processo prossegue normalmente. No terceiro grupo temos as regras para um processo afectado por uma restrição. Neste grupo têm particular interesse as regras (ν_{bo}) e $(\nu_{bo\,in})$. Na primeira há uma extrusão do nome restrito ou seja uma saída ligada, e na segunda são tratados, em simultâneo, o caso de uma entrada e o caso de uma extrusão de um nome restrito dentro de uma outra restrição. Nos dois grupos seguintes são tratados os casos dos processos compostos por soma e por composição paralela, em que as transições são inferidas a partir das transições dos processos componentes. No último grupo é tratada a composição paralela com sincronização entre as partes componentes. Na regra $(sync)$, o processo P comunica o nome x, pelo canal a, ao processo Q e o parâmetro formal de Q é substituído pelo nome recebido. Em $(sync_\nu)$, o processo P está a enviar um nome que era restrito por isso a restrição passa a abranger ambos os processos envolvidos na comunicação, pois agora o nome privado passa a estar no âmbito dos dois processos. As regras onde aparece $\hat{a}$ dão conta de duas

situações distintas, conforme o valor que $\hat{a}$ assume. Para $\hat{a} = a$ é recebido um nome pelo canal a e prossegue-se com uma função que depende do nome recebido. Para $\hat{a} = \overline{a}$ é enviado um nome ligado pelo canal a e prossegue-se com uma função até que a restrição se expanda até ao processo que recebe o nome restrito. Note-se que na conclusão de (ν_{boin}) e de (com_{boin}) não impomos que z seja diferente de x porque a própria definição de substituição garante que se $x = z$ então, antes de efectuar a substituição propriamente dita, é efectuada ao processo uma conversão-α que troca o nome x ligado por um nome fresco. Saliente-se ainda que não existem regras específicas para a replicação; todas as transições da replicação são obtidas através da regra da congruência (*cong*). Historicamente as primeiras apresentações do cálculo-π não utilizavam a congruência estrutural, com a vantagem de tornarem mais claras algumas demonstrações por indução. Actualmente a maioria das apresentações usa alguma forma de congruência, pelo menos a congruência-α, uma vez que isso torna as definições mais compactas e transparentes. Só para o caso da replicação, se quiséssemos prescindir da congruência, teríamos de acrescentar ao sistema cinco regras. Ora vejamos,

Replicação:

$$(\text{repl}_{\tau})\ \frac{P \xrightarrow{\tau} Q}{!P \xrightarrow{\tau} !P|Q}$$

$$(\text{repl}_{out})\ \frac{P \xrightarrow{\overline{a}} < x > Q}{!P \xrightarrow{\overline{a}} < x > (!P|Q)}$$

$$(\text{repl}_{boin})\ \frac{P \xrightarrow{\hat{a}} \lambda_z Q\{z/w\}}{!P \xrightarrow{\hat{a}} \lambda_z (!P|Q)\{z/w\}}\ \ w \notin lv(!P),\ \ \hat{a} \in \{a, \overline{a}\}$$

$$(\text{repl-sync})\ \frac{P \xrightarrow{\overline{a}} < x > Q \quad P \xrightarrow{a} \lambda_z Q'\{z/w\}}{!P \xrightarrow{\tau} !P|Q|Q'\{x/w\}}$$

$$(\text{repl}_{sync\,\nu})\ \frac{P \xrightarrow{\overline{a}} \lambda_z Q\{z/w\} \quad P \xrightarrow{a} \lambda_z Q'\{z/w\}}{!P \xrightarrow{\tau} !P|(\nu x)(Q\{x/w\}|Q'\{x/w\})}\ x \notin (lv(Q)\cup lv(Q'))-\{w\}$$

As duas primeiras regras correspondem à execução de uma acção interna e ao envio de um nome para o exterior, respectivamente. A terceira regra, conforme o valor que $\hat{a}$ assume, resolve o caso da recepção de um nome ou o do envio de um nome ligado. A regra (repl_{sync}) corresponde ao caso em que o processo pode executar duas transições complementares, o envio e a recepção de um nome pelo mesmo canal, e logo, ao replicar-se, de forma a obtermos $!P|P|P$ as duas "cópias" de P podem sincronizar-se e comunicar. A regra ($\text{repl}_{sync\,\nu}$) corresponde a um caso semelhante ao anterior mas em que o nome enviado aparece ligado em P.

Segue-se uma propriedade que justifica uma certa liberdade na escolha da função que representa a entrada de um nome ou a saída de um nome ligado.

Propriedade 4.3.7 *Seja $f = \lambda_z Q\{z/w\}$ e $v \notin lv(Q)$ então $f = \lambda_z Q^{(vw)}\{z/v\}$*

Demonstração Ora $\{z/v\} \circ (vw) = \{z/w,\, w/v\}$ mas como $v \notin lv(Q)$ tem-se $Q\{z/w\} = Q\{z/w,\, w/v\}$. □

Os dois lemas que se seguem contêm resultados que nos vão permitir simplificar algumas demonstrações mais adiante.

Lema 4.3.8 *Seja Q um conjunto nominal. Um sistema-pn $\xi : Q \to F(Q)$ é um morfismo de conjuntos nominais se e só se*

1. *Se $q \xrightarrow{\alpha} \delta$ então $q^\sigma \xrightarrow{\alpha^\sigma} \delta^\sigma$;*
2. *Se $q^\sigma \xrightarrow{\alpha^\sigma} \eta$ então existe δ tal que $q \xrightarrow{\alpha} \delta$ e $\delta^\sigma = \eta$.*

Demonstração Atendendo a que o functor F é a soma de quatro conjuntos vamos fazer a demonstração para o caso em que a transição envolve o primeiro conjunto, $\xi_\tau : Q \to \mathcal{P}_{fin}(Q)$, e para o caso em que a transição envolve o

último conjunto, $\xi_{in} : Q \to \mathcal{P}_{fin}(\mathcal{N} \times Q^{\mathcal{N}})$. Os outros casos são semelhantes. Comecemos por supor que $\xi_\tau : Q \to \mathcal{P}_{fin}(Q)$ é um morfismo e vamos mostrar a primeira implicação.

1. Se $q \xrightarrow{\tau} p$ então $p \in \xi_\tau(q)$, logo $p^\sigma \in \xi_\tau(q)^\sigma$. Como, por hipótese, ξ_τ é um morfismo, vem $p^\sigma \in \xi_\tau(q^\sigma)$ e portanto $q^\sigma \xrightarrow{\tau^\sigma} \delta^\sigma$.

2. Se $q^\sigma \xrightarrow{\tau} r$ então $r \in \xi_\tau(q^\sigma)$, mas, como por hipótese, ξ_τ é um morfismo, vem $\xi_\tau(q^\sigma) = \xi_\tau(q)^\sigma = \{p^\sigma : p \in \xi_\tau(q)\} = \{p^\sigma : q \xrightarrow{\tau} p\}$ e logo existe p tal que $q \xrightarrow{\tau} p \,\&\, r = p^\sigma$.

Para a implicação reciproca, temos de verificar que $\xi_\tau(q^\sigma) = \xi_\tau(q)^\sigma$. Ora $\xi_\tau(q^\sigma) = \{r : q^\sigma \xrightarrow{\tau} r\}$ que, por *1* e *2*, é igual a $\{p^\sigma : q \xrightarrow{\tau} p\} = \{p : q \xrightarrow{\tau} p\}^\sigma = \xi_\tau(q)^\sigma$, como se pretendia.
Passemos à demonstração para as transições que envolvem o quarto conjunto. Supondo que $\xi_{in} : Q \to \mathcal{P}_{fin}(\mathcal{N} \times Q^{\mathcal{N}})$ é um morfismo tem-se

1. Se $q \xrightarrow{a} t$ então $(a, t) \in \xi_{in}(q)$, logo $(a, t)^\sigma \in \xi_{in}(q)^\sigma$, isto é $(a^\sigma, t^\sigma) \in \xi_{in}(q)^\sigma$. Como, por hipótese, ξ_{in} é um morfismo, vem $(a^\sigma, t^\sigma) \in \xi_{in}(q^\sigma)$ e portanto $q^\sigma \xrightarrow{a^\sigma} t^\sigma$.

2. Se $q^\sigma \xrightarrow{b} f$ então $(b, f) \in \xi_{in}(q^\sigma)$, mas, como por hipótese, ξ_{in} é um morfismo, vem $\xi_\tau(q^\sigma) = \xi_\tau(q)^\sigma = \{(a^\sigma, t^\sigma) : q \xrightarrow{a} t\}$ e logo existe (a, t) tal que $q \xrightarrow{a} t \,\&\, b = a^\sigma,\ f = t^\sigma$, como se pretendia.

Para a implicação reciproca, vem
$\xi_{in}(q^\sigma) = \{(b, f) : q^\sigma \xrightarrow{b} f\}$ que por *1* e *2* é igual a $\{(a, t)^\sigma : q \xrightarrow{a} t\} = \{(a, t) : q \xrightarrow{a} t\}^\sigma = \xi_{in}(q)^\sigma$. □

Lema 4.3.9 *Seja Q um conjunto nominal. Para garantir que um sistema-pn $\xi : Q \to F(Q)$ é um morfismo de conjuntos nominais basta provar que*

$$\textit{se } P \xrightarrow{\alpha} \delta \textit{ então } P^\sigma \xrightarrow{\alpha^\sigma} \delta^\sigma.$$

Demonstração Para garantir que $\xi : Q \to F(Q)$ é um morfismo de conjuntos nominais temos de provar que:

(a) Se $q \xrightarrow{\alpha} \delta$ então $q^\sigma \xrightarrow{\alpha^\sigma} \delta^\sigma$.

(b) Se $q^\sigma \xrightarrow{\alpha^\sigma} \eta$ então existe δ tal que $q \xrightarrow{\alpha} \delta$ e $\delta^\sigma = \eta$.

Mas, se provarmos (a), para provar (b) basta atendermos a que, se $q^\sigma \xrightarrow{\alpha^\sigma} \eta$ então, por (a) vem $q \xrightarrow{\alpha} \eta^{\sigma^{-1}}$ pelo que fazendo $\delta = \eta^{\sigma^{-1}}$ tem-se $\delta^\sigma = \eta$ como pretendíamos. □

Lema 4.3.10 *γ é um morfismo de conjuntos-*Π.

Demonstração Atendendo aos dois lemas anteriores temos de mostrar que:

se $q \xrightarrow{\alpha} \delta$ então $q^\sigma \xrightarrow{\alpha^\sigma} \delta^\sigma$.

Comecemos por demonstrar o resultado para os prefixos. Se $a(x).P \xrightarrow{a} \lambda_z P\{z/x\}$ também queremos ter $(a(x).P)^\sigma \xrightarrow{a^\sigma} (\lambda_z P\{z/x\})^\sigma$ mas, $(\lambda_z P\{z/x\})^\sigma = \lambda_z (P\{z^{\sigma^{-1}}/x\})^\sigma = \lambda_z P^\sigma\{z/x^\sigma\}$ e portanto o que queremos é ter $a^\sigma(x^\sigma).P^\sigma \xrightarrow{a^\sigma} \lambda_z P^\sigma\{z/x^\sigma\}$ que é uma instância do axioma (in). Se tivermos $\overline{a}\langle x\rangle.P \xrightarrow{\overline{a}} (a)P$ então também queremos ter $(\overline{a}\langle x\rangle.P)^\sigma \xrightarrow{\overline{a^\sigma}} ((a)P)^\sigma \Leftrightarrow \overline{a^\sigma}\langle x^\sigma\rangle.P^\sigma \xrightarrow{\overline{a^\sigma}} (a^\sigma)P^\sigma$ que é uma instância do axioma (out). De modo idêntico, se tivermos $\tau.P \xrightarrow{\tau} P$ também se tem $(\tau.P)^\sigma \xrightarrow{\tau^\sigma} P^\sigma \Leftrightarrow \tau.P^\sigma \xrightarrow{\tau} P^\sigma$ que é uma instância do axioma (tau).

(*sum*) Se $P_1 + P_2 \xrightarrow{\alpha} \delta$ com , suponhamos, $P_1 \xrightarrow{\alpha} \delta$, por hipótese de indução $P_1^\sigma \xrightarrow{\alpha^\sigma} \delta^\sigma$ e logo $P_1^\sigma + P_2^\sigma \xrightarrow{\alpha^\sigma} \delta^\sigma$. Se fosse $P_2 \xrightarrow{\alpha} \delta$ a demonstração era idêntica.

(ν_{bo}) Se tivermos $(\nu x)P \xrightarrow{\overline{a}} \lambda_z P'\{z/x\}$ com $P \xrightarrow{\overline{a}} \langle x\rangle P'$ e $x \neq a$ por hipótese de indução $P^\sigma \xrightarrow{\overline{a^\sigma}} (\langle x\rangle P')^\sigma \Rightarrow P^\sigma \xrightarrow{\overline{a^\sigma}} \langle x^\sigma\rangle P'^\sigma$ e, como também $x^\sigma \neq a^\sigma$, por (ν_{bo}) vem

$(\nu x^\sigma)P^\sigma \xrightarrow{\overline{a^\sigma}} \lambda_z P'^\sigma\{z/x^\sigma\}$

$\Rightarrow ((\nu x)P)^\sigma \xrightarrow{\overline{a^\sigma}} \lambda_z (P'\{z^{\sigma^{-1}}/x\})^\sigma$

$\Rightarrow ((\nu x)P)^\sigma \xrightarrow{\overline{a^\sigma}} (\lambda_z P'\{z/x\})^\sigma$, como se pretendia.

(ν_{boin}) Se tivermos $(\nu x)P \xrightarrow{\hat{a}} \lambda_z((\nu x)Q)\{z/w\}$ com $P \xrightarrow{\hat{a}} \lambda_z Q\{z/w\}$ e $x \neq a, x \neq w$ por hipótese de indução $P^\sigma \xrightarrow{\hat{a^\sigma}} (\lambda_z Q\{z/w\})^\sigma \Rightarrow P^\sigma \xrightarrow{\hat{a^\sigma}} \lambda_z(Q\{z^{\sigma^{-1}}/w\}) \Rightarrow P^\sigma \xrightarrow{\hat{a^\sigma}} \lambda_z Q^\sigma\{z/w^\sigma\})$ e, como também $x^\sigma \neq a^\sigma$ e $x^\sigma \neq w^\sigma$, por (ν_{boin}) vem

$(\nu x^\sigma)P^\sigma \xrightarrow{\hat{a^\sigma}} \lambda_z((\nu x^\sigma)Q^\sigma)\{z/w^\sigma\}$

$\Rightarrow ((\nu x)P)^\sigma \xrightarrow{\hat{a^\sigma}} \lambda_z(((\nu x)Q)\{z^{\sigma^{-1}}/w\})^\sigma$

$\Rightarrow ((\nu x)P)^\sigma \xrightarrow{\hat{a^\sigma}} (\lambda_z((\nu x)Q)\{z/w\})^\sigma$, como se pretendia.

($comp_\tau$) Se tivermos $P|Q \xrightarrow{\tau} P'|Q$ com $P \xrightarrow{\tau} P'$, por hipótese de indução, $P^\sigma \xrightarrow{\tau^\sigma} P'^\sigma$ e por $comp_\tau$ obtemos $P^\sigma|Q^\sigma \xrightarrow{\tau} P'^\sigma|Q^\sigma$ que é equivalente a $(P|Q)^\sigma \xrightarrow{\tau^\sigma} (P'|Q)^\sigma$ que é a transição que pretendemos.

($sync$) Se tivermos $P|Q \xrightarrow{\tau} P'|\lambda_z Q'\{z/w\}$ com $P \xrightarrow{\overline{a}} \langle x\rangle P'$ e $Q \xrightarrow{a} \lambda_z Q'\{z/w\}$, por hipótese de indução, $P^\sigma \xrightarrow{\overline{a^\sigma}} (x^\sigma)P'^\sigma$ e $Q^\sigma \xrightarrow{a^\sigma} (\lambda_z Q'\{z/w\})^\sigma$ que é equivalente a $Q^\sigma \xrightarrow{a^\sigma} \lambda_z Q'^\sigma\{z/w^\sigma\}$ e por $sync$ obtemos $P^\sigma|Q^\sigma \xrightarrow{\tau} P'^\sigma|Q'^\sigma\{x^\sigma/w^\sigma\}$ que por sua vez é equivalente a $(P|Q)^\sigma \xrightarrow{\tau^\sigma} (P'|Q'\{x/w\})^\sigma$ que é a transição que pretendemos.

($sync_\nu$) Se tivermos $P|Q \xrightarrow{\tau} (\nu x)(P'\{x/w\}|Q'\{x/v\})$ com $P \xrightarrow{\overline{a}} \lambda_z P'\{z/w\}$ e $Q \xrightarrow{a} \lambda_w Q'\{z/w\}$ para $x \notin (lv(P') - \{w\}) \cup (lv(Q') - \{w\})$, por hipótese de indução, $P^\sigma \xrightarrow{\overline{a^\sigma}} (\lambda_z P'\{z/w\})^\sigma$ e $Q^\sigma \xrightarrow{a^\sigma} (\lambda_w Q'\{z/w\})^\sigma$ que é equivalente a $P^\sigma \xrightarrow{\overline{a^\sigma}} \lambda_z P'^\sigma\{z/w^\sigma\}$ e $Q^\sigma \xrightarrow{a^\sigma} \lambda_z Q'^\sigma\{z/w^\sigma\}$
e por $sync$ obtemos $P^\sigma|Q^\sigma \xrightarrow{\tau} (\nu x^\sigma)(P'^\sigma\{x^\sigma/w^\sigma\}|Q'^\sigma\{x^\sigma/w^\sigma\})$ que é equivalente a $(P|Q)^\sigma \xrightarrow{\tau^\sigma} ((\nu x)(P'\{x/w\}|Q'\{x/w\}))^\sigma$ que é a transição que pretendemos.

Os restantes casos demonstram-se por raciocínios semelhantes. □

Estamos agora em condições de definir a semântica operacional do Cálculo-π como o único morfismo definido por γ na coálgebra final.

Definição 4.3.11 (Semântica operacional) Seja $\nu : \mathsf{Proc} \to \mathbb{D}$ o único morfismo definido por $\gamma : Proc \to F(Proc)$ na coálgebra final $(\mathbb{D}, \chi)$. A

semântica operacional de um processo $P \in \mathsf{Proc}$ é dada por

$$\mathcal{O}(P) = \nu(P).$$

□

4.4 Semântica Denotacional

Na semântica denotacional define-se o valor que cada processo denota no domínio semântico mas de forma composicional, isto é, de forma a que o significado de um processo seja composto a partir dos significados das partes que o constituem. Para o caso denotacional o domínio semântico é ainda $\mathbb{D}$. Definimos em primeiro lugar a semântica denotacional fechada e depois, à custa desta, definimos a semântica aberta. No caso fechado a semântica é definida à custa de funções sobre entidades semânticas. Estas funções são definidas de dois modos distintos, conforme o formato das entidades. Os casos mais simples – que correspondem ao processo nulo, ao prefixo e à soma – permitem uma abordagem mais directa e a respectiva semântica é obtida à custa de χ e da sua inversa. Quanto aos outros casos – composição paralela, restrição e replicação – a respectiva semântica denotacional é obtida recorrendo a um sistema de transições sobre um domínio apropriado de forma análoga à utilizada para obter a semântica operacional. Seria fácil ampliar o sistema de transições (e modificar o domínio) de forma a este incluir também os casos mais simples que seriam tratados do mesmo modo que os restantes. Mais complicado seria tratar a composição paralela, a restrição e a replicação à custa de χ e da sua inversa pois concluímos que iríamos obter expressões muito elaboradas sobre as quais seria difícil estabelecer resultados. Optou-se por esta abordagem mista que permite aplicar as técnicas que melhor se adaptam a cada um dos dois grupos.

Vamos começar pela definição de algumas funções e resultados auxiliares.

Definição 4.4.1 (Operações auxiliares) Definem-se as seguintes operações auxiliares:

$out : \mathcal{N} \times \mathcal{N} \times \mathbb{D} \to \mathbb{D}$
$\quad out(\overline{a}, x, T) = \chi^{-1}(\{(\overline{a}, x, T)\});$

$in : \mathcal{N} \times \mathbb{D}^{\mathcal{N}} \to \mathbb{D}$
$\quad in(a, f) = \chi^{-1}(\{(a, f)\});$

$sil : \mathbb{D} \to \mathbb{D}$
$\quad sil(T) = \chi^{-1}(\{T\});$

$null : \mathbb{D}^0 \to \mathbb{D}$
$\quad null = \chi^{-1}(\emptyset);$

$sum : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$
$\quad sum(T, U) = \chi^{-1}(\chi(T) \cup \chi(U)).$

□

Lema 4.4.2 *As operações anteriormente apresentadas estão bem definidas.*

Demonstração A função *out* está bem definida pois dado $(\overline{a}, x, T) \in \mathcal{N} \times \mathcal{N} \times \mathbb{D}$, temos $\{(\overline{a}, x, T)\} \in \mathcal{P}_{fin}(\mathcal{N} \times \mathcal{N} \times \mathbb{D})$ e logo $\chi^{-1}(\{(\overline{a}, x, T)\})$ está bem definido e é um elemento de $\mathbb{D}$, pois como χ é final é um isomorfismo de $\mathbb{D}$ em $F(\mathbb{D})$. Por raciocínio semelhante conclui-se que também *in* e *sil* estão bem definidas. Como $\emptyset \in F(\mathbb{D})$, $\chi^{-1}(\emptyset)$ está definido e é um elemento de $\mathbb{D}$ ($\chi : \mathbb{D} \cong F(\mathbb{D})$) e tem-se $\chi^{-1}(\emptyset) = (\emptyset_n)$ onde $(\emptyset_n)$ denota a sucessão $\emptyset[n] = \emptyset,\ \forall_n \geq 0$, o que garante que *null* está bem definida. Dados $T, U \in \mathbb{D}$, tem-se $\chi(T)$ e $\chi(U)$ elementos de $F(\mathbb{D})$ e a sua união é ainda um elemento de $F(\mathbb{D})$, pelo que $\chi^{-1}(\chi(T) \cup \chi(U))$ está bem definida e o resultado é um elemento de $\mathbb{D}$ o que garante que *sum* está bem definida. □

Vamos agora introduzir um domínio auxiliar que será utilizado para a definição do sistema de transições a partir do qual se definirão, em $\mathbb{D}$, as operações de composição paralela, restrição e replicação.

Definição 4.4.3 (Domínio auxiliar $\mathcal{Q}$) Seja $\mathcal{Q}_0 = \mathbb{D}$ e $\mathcal{Q}_{n+1} = \mathcal{Q}_n + (\mathcal{Q}_n \times \mathcal{Q}_n) + \mathcal{Q}_n + \mathcal{Q}_n^{\mathcal{N}}$. Define-se $\mathcal{Q} = \cup_{n\geq 0}\mathcal{Q}_n$. □

Para todo o n, $\mathcal{Q}_n$ é um conjunto nominal, pois $\mathcal{Q}_0 = \mathbb{D}$ é um conjunto nominal e, fazendo $Z(Q) = Q + (Q \times Q) + Q + Q^{\mathcal{N}}$, tem-se $\mathcal{Q}_{n+1} = Z(\mathcal{Q}_n)$ onde os functores que compõem Z preservam conjuntos nominais. Portanto $\mathcal{Q} = \cup_{n\geq 0}\mathcal{Q}_n$ é um conjunto nominal. Mais precisamente, se $in : \mathcal{Q}_n \to \mathcal{Q}_{n+1}$ for a inclusão então $Z(in)$ também é a inclusão, isto implica que a acção de qualquer permutação σ em $Z(\mathcal{Q}_n)$ é a restrição da acção da mesma permutação σ em $Z(\mathcal{Q}_{n+1})$. Podemos pois concluir que a união dos $\mathcal{Q}_n$ continua a ser um conjunto nominal e a acção num elemento δ de $\mathcal{Q}$ é a extensão da acção em $Z(\mathcal{Q}_n)$ para algum $\mathcal{Q}_n$ tal que $\delta \in \mathcal{Q}_n$

Iremos utilizar a seguinte notação para os elementos de $\mathcal{Q}_{n+1}$:

$(0, q)$ é denotado por q
$(1, q, q')$ é denotado por $q|q'$
$(2, q)$ é denotado por $!q$
$(3, e)$ é denotado por e

O functor $Z(Q) = Q + (Q \times Q) + Q + Q^{\mathcal{N}}$ é monótono, isto é, se $Q \subseteq Q'$ então $Z(Q) \subseteq Z(Q')$, nos moldes descritos na secção 2.2.6.

Tem-se $\mathcal{Q}_0 \subseteq \mathcal{Q}_1$ porque $\mathcal{Q}_0 = \mathbb{D}$ e $\mathcal{Q}_1 = \mathbb{D} + (\mathbb{D} \times \mathbb{D}) + \mathbb{D} + \mathbb{D}^{\mathcal{N}}$ e, mais geralmente, $\mathcal{Q}_n \subseteq \mathcal{Q}_{n+1}$. A notação $q|q'$ introduzida para $\mathcal{Q}_{n+1}$ pode ser estendida a $\mathcal{Q}$ do seguinte modo: se $q \in \mathcal{Q}_n$ e $q' \in \mathcal{Q}_k$, tomamos m igual ao maior de entre n e k; então $q|q' \in \mathcal{Q}_m \times \mathcal{Q}_m \subseteq \mathcal{Q}_{m+1} \subseteq \mathcal{Q}$. Analogamente pode-se comparar $!q$ e $!q'$, ou e e e', com $q, q', e, e' \in \mathcal{Q}$ porque eles podem ser pensados como pertencendo ao mesmo $\mathcal{Q}_n$.

Definição 4.4.4 (Relação de congruência em $\mathcal{Q}$) Uma relação binária em $\mathcal{Q}$ é uma relação de congruência se for uma relação de equivalência e para todo o $q_1, q_2, q_1', q_2', e, e' \in \mathcal{Q}$ se tem

$$q_1 \cong q_1',\ q_2 \cong q_2' \Rightarrow q_1|q_2 \cong q_1'|q_2'$$
$$sup(e) = sup(e') \,\&\, \forall_x\ e(x) \cong e'(x) \Rightarrow e \cong e'$$
$$q_1 \cong q_1' \Rightarrow !q_1 \cong !q_1'$$

□

Definição 4.4.5 (Congruência semântica) Consideremos em $\mathcal{Q}$ a menor relação de congruência, $\equiv$, que satisfaz

$$\begin{array}{ll} (par\ nul) & q|\mathsf{0} \equiv q; \\ (par\ comu) & q|q' \equiv q'|q; \\ (par\ assoc) & q|(q'|q'') \equiv (q|q')|q''; \\ (res\ par) & q|e \equiv \lambda_x(q|e(x)); \\ (repl) & !q \equiv !q|q. \end{array}$$

□

Note-se que esta relação quando restringida a $\mathcal{Q}_0(= \mathbb{D})$ é a identidade. Vamos estender a relação de congruência para o functor F_0, definido em 4.2.1, agora no contexto denotacional.

Definição 4.4.6 (Congruência em $F_0(\mathcal{Q})$) Sejam $\delta, \delta' \in F_0(\mathcal{Q})$. Tem-se δ congruente com δ', $\delta \equiv \delta'$ se:

$$\begin{array}{ll} (cong\ \tau) & \delta = q \Rightarrow \delta' = q' \,\&\, q \equiv q'; \\ (cong\ out) & \delta = (\overline{a}, x, q) \Rightarrow \delta' = (\overline{a}, x, q') \,\&\, q \equiv q'; \\ (cong\ bo) & \delta = (\overline{a}, e) \Rightarrow \delta' = (\overline{a}, e'),\ sup(e) = sup(e'), \forall_x\ e(x) \equiv e'(x); \\ (cong\ in) & \delta = (a, t) \Rightarrow \delta' = (a, t'),\ sup(t) = sup(t'), \forall_x\ t(x) \equiv t'(x). \end{array}$$

□

Definição 4.4.7 (Sistema de transições em $\mathcal{Q}$) Consideremos o sistema de transições etiquetadas, $\eta : \mathcal{Q} \to F(\mathcal{Q})$, cuja especificação é definida pelas seguintes regras e axiomas:

Base

$$(base)\ \frac{q \xrightarrow{\alpha}_{\chi} \delta}{q \xrightarrow{\alpha}_{\eta} \delta}$$

Congruência

$$(cong)\ \frac{q_1 \equiv q_2 \qquad q_2 \xrightarrow{\alpha}_{\eta} \delta_2 \qquad \delta_2 \equiv \delta_1}{q_1 \xrightarrow{\alpha}_{\eta} \delta_1}$$

Composição

$$(par_{\tau})\ \frac{q \xrightarrow{\tau}_{\eta} q''}{q|q' \xrightarrow{\tau}_{\eta} q''|q'}$$

$$(par_{out})\ \frac{q \xrightarrow{\overline{a}}_{\eta} \langle x \rangle q''}{q|q' \xrightarrow{\overline{a}}_{\eta} \langle x \rangle (q''|q')}$$

$$(par_{boin})\ \frac{q \xrightarrow{\hat{a}}_{\eta} e}{q|q' \xrightarrow{\hat{a}}_{\eta} \lambda_x(e(x)|q')} \qquad \hat{a} \in \{a, \overline{a}\}$$

Comunicação

$$(com)\ \frac{q \xrightarrow{\overline{a}}_{\eta} \langle x \rangle q'' \qquad\qquad q' \xrightarrow{a}_{\eta} t}{q|q' \xrightarrow{\tau}_{\eta} q''|t(x)}$$

$$(com_{res})\ \frac{q \xrightarrow{\overline{a}}_{\eta} e \qquad\qquad q' \xrightarrow{a}_{\eta} t}{q|q' \xrightarrow{\tau}_{\eta} \lambda_x(e(x)|t(x))}$$

Restrição

$$(res_{\tau})\ \frac{e(x) \xrightarrow{\tau}_{\eta} e'(x)}{e \xrightarrow{\tau}_{\eta} e'},\ x \notin sup(e)$$

$$(res_{out})\ \frac{e(x) \xrightarrow{\overline{a}}_{\eta} \langle y\rangle e'(x)}{e \xrightarrow{\overline{a}}_{\eta} \langle y\rangle e'},\ x \notin sup(e),\ \ x \neq a,\ x \neq y$$

$$(res_{bo})\ \frac{e(x) \xrightarrow{\overline{a}}_{\eta} \langle x\rangle e'(x)}{e \xrightarrow{\overline{a}}_{\eta} e'},\ \ x \notin sup(e),\ x \neq a$$

$$(res_{bo\,in})\ \frac{e(x) \xrightarrow{\hat{a}}_{\eta} E(x)}{e \xrightarrow{\hat{a}}_{\eta} \lambda_z\lambda_x E(x)(z)},\ \ \hat{a} \in \{a, \overline{a}\},\ x \notin sup(e) \cup sup(E),\ \ x \neq a.$$

□

No primeiro grupo está a regra (base), aonde todas as outras acabam por conduzir, e que estabelece que as transições dos elementos de $\mathbb{D}$ são as herdadas da coálgebra final $(\mathbb{D}, \chi)$. O segundo grupo estabelece que elementos congruentes entre si têm transições com a mesma etiqueta para elementos também congruentes entre si. No terceiro grupo estão as regras para a "composição paralela" nos casos em que há evolução de apenas uma das componentes. Neste grupo a regra $(par_{bo\,in})$ trata em simultâneo os casos da entrada de um nome, quando $\hat{a} = a$, e da saída ligada, quando $\hat{a} = \overline{a}$. Note-se que, na expressão $\lambda_x(e(x)|q')$ da conclusão, como o nome x não aparece explícito em q' ele não ocorre na expressão ou, caso ocorra, funciona como uma constante para a função, isto é, ao calcular a valor da função para um nome, digamos z, apenas as ocorrências do nome x de $e(x)$ é que são instanciadas com z, mantendo-se as ocorrências de x em q' inalteradas. Esta é uma convenção que vai ser utilizada ao logo de todo o trabalho para esse tipo de funções: os parâmetros de uma função são sempre explicitamente indicados, para evitar ambiguidades. No grupo seguinte tratam-se os casos

das composição paralela em que há comunicação entre os componentes. A primeira regra deste grupo refere-se à situação simples de mera comunicação de um nome por um canal comum e a segunda regra refere-se ao caso em que há uma extrusão de um nome restrito que se traduz por meio de uma função. Em último, vem o grupo que trata a restrição. Denotacionalmente a restrição de uma variável x é interpretada como uma função que depende do nome restrito. As duas primeiras regras deste grupo correspondem a uma acção interna e a uma saída, respectivamente. A terceira regra trata uma extrusão do nome x. A última regra trata, em simultâneo, o caso da entrada de um nome (se $\hat{a} = a$) e da extrusão de um nome x após a extrusão de um nome z (caso $\hat{a} = \overline{a}$). Aqui denotamos por E uma função de "nível" superior a e, isto é, se pensarmos em e como uma função de $\mathcal{N}$ em $\mathcal{Q}_n$ então E é uma função de $\mathcal{N}$ em $\mathcal{Q}_{n+1}$.

Vamos seguidamente mostrar que, definida deste modo, η constitui um morfismo de conjuntos nominais.

Lema 4.4.8 *$\eta : \mathcal{Q} \to F(\mathcal{Q})$ é um morfismo de conjuntos nominais.*

Demonstração Temos de mostrar que se $q \xrightarrow{\alpha}_\eta \delta$ então $q^\sigma \xrightarrow{\alpha^\sigma}_\eta \delta^\sigma$.
Para as transições obtidas da regra *base* o resultado é imediato pelo facto de χ ser morfismo. Vamos demonstrar o resultado para as transições obtidas por algumas das outras regras. Para as restantes regras a demonstração faz-se de modo semelhante.

(par_τ) Se $q|q' \xrightarrow{\tau}_\eta q''|q'$ com $q \xrightarrow{\tau}_\eta q''$, por indução estrutural tem-se $q^\sigma \xrightarrow{\tau^\sigma}_\eta q''^\sigma$ e pela regra (par_τ) obtém-se $q^\sigma|q'^\sigma \xrightarrow{\tau}{}^{\sigma}_{\eta} q''^\sigma|q'^\sigma$ que é equivalente a $(q|q')^\sigma \xrightarrow{\tau^\sigma}_\eta (q''|q')^\sigma$ que é a transição pretendida.

(par_{boin}) Se $q|q' \xrightarrow{\hat{a}}_\eta \lambda_x(e(x)|q')$ com $q \xrightarrow{\hat{a}}_\eta e$ por indução estrutural tem-se $q^\sigma \xrightarrow{\hat{a}^\sigma}_\eta e^\sigma$ e pela regra (par_{boin}) obtém-se $q^\sigma|q'^\sigma \xrightarrow{\hat{a}^\sigma}_\eta \lambda_x(e^\sigma(x)|q'^\sigma)$ que é equivalente a $(q|q')^\sigma \xrightarrow{\hat{a}^\sigma}_\eta \lambda_x(e(x^{\sigma^{-1}})|q')^\sigma$, por sua vez, equivalente a $(q|q')^\sigma \xrightarrow{\hat{a}^\sigma}_\eta (\lambda_x(e(x)|q'))^\sigma$ que é a transição pretendida.

(com_{res}) Se $q|q' \xrightarrow{\tau}_\eta \lambda_x(e(x)|t(x))$ com $q \xrightarrow{\overline{a}}_\eta e \,\&\, q' \xrightarrow{a}_\eta t$ por indução tem-se $q^\sigma \xrightarrow{\overline{a^\sigma}}_\eta e^\sigma \,\&\, q'^\sigma \xrightarrow{a^\sigma}_\eta t^\sigma$ e pela regra (com_{res}) obtém-se $q^\sigma|q'^\sigma \xrightarrow{\tau^\sigma}_\eta$

$\lambda_x(e^\sigma(x)|t^\sigma(x))$ que é equivalente a $(q|q')^\sigma \xrightarrow{\tau^\sigma}_\eta \lambda_x(e(x^{\sigma^{-1}})|t(x^{\sigma^{-1}}))^\sigma$ por sua vez equivalente a $(q|q')^\sigma \xrightarrow{\tau^\sigma}_\eta (\lambda_x(e(x)|t(x)))^\sigma$ que é a transição pretendida.

(res_τ) Se tivermos $e \xrightarrow{\tau}_\eta e'$ com $e(x) \xrightarrow{\tau}_\eta e'(x)$ para $x \notin sup(e)$, por hipótese de indução tem-se $e(x)^\sigma \xrightarrow{\tau^\sigma}_\eta e'(x)^\sigma$ que implica $e^\sigma(x^\sigma) \xrightarrow{\tau}_\eta e'^\sigma(x^\sigma)$ e além disso como $x \notin sup(e)$ então $x^\sigma \notin sup(e)^\sigma$ e, como e é finitamente suportada, $x^\sigma \notin sup(e^\sigma)$. Estamos em condições de aplicar (res_τ) e obtemos $e^\sigma \xrightarrow{\tau}_\eta e'^\sigma$ o que implica $e^\sigma \xrightarrow{\tau^\sigma}_\eta e'^\sigma$.

(res_{bo}) Se tivermos $e \xrightarrow{\overline{a}}_\eta e'$ com $e(x) \xrightarrow{\overline{a}}_\chi \langle x\rangle e'(x)$ para $x \notin sup(e)$ e $x \neq a$, por hipótese de indução $e(x)^\sigma \xrightarrow{\overline{a}^\sigma}_\eta (\langle x\rangle e'(x))^\sigma$ que implica $e^\sigma(x^\sigma) \xrightarrow{\overline{a}^\sigma}_\eta \langle x\rangle^\sigma e'(x)^\sigma$ que por sua vez implica $e^\sigma(x^\sigma) \xrightarrow{\overline{a}^\sigma}_\eta \langle x^\sigma\rangle e'^\sigma(x^\sigma)$ e como $x^\sigma \notin sup(e^\sigma)$ e $x^\sigma \neq a^\sigma$, por (res_{bo}) vem $e^\sigma \xrightarrow{\overline{a}^\sigma}_\eta e'^\sigma$.

$(res_{bo\,in})$ Se $e \xrightarrow{\hat{a}}_\eta \lambda_z\lambda_x E(x)(z)$ com $e(x) \xrightarrow{\hat{a}}_\eta E(x)$ para $x \notin sup(e) \cup sup(E)$, $a \neq x$, $\hat{a} \in \{a, \overline{a}\}$ logo $x^\sigma \notin sup(e^\sigma) \cup sup(E^\sigma)$ e $a^\sigma \neq x^\sigma$. Por hipótese de indução $e(x)^\sigma \xrightarrow{\hat{a}^\sigma}_\eta E(x)^\sigma$ o que implica $e^\sigma(x^\sigma) \xrightarrow{\hat{a}^\sigma}_\eta E^\sigma(x^\sigma)$, donde por $(res_{bo\,in})$, $e^\sigma \xrightarrow{\hat{a}^\sigma}_\eta \lambda_z\lambda_{x^\sigma} E^\sigma(x^\sigma)(z)$ e logo $e^\sigma \xrightarrow{\hat{a}^\sigma}_\eta \lambda_z\lambda_x E^\sigma(x)(z)$, porque σ é uma bijecção. Este é já o resultado pretendido pois

$$\begin{aligned}(\lambda_z\lambda_x E(x)(z))^\sigma &= \lambda_z(\lambda_x E(x)(z^{\sigma^{-1}}))^\sigma \\ &= \lambda_z E(x^{\sigma^{-1}})(z^{\sigma^{-1}})^\sigma \\ &= \lambda_z\lambda_x(E(x^{\sigma^{-1}}))^\sigma(z) \\ &= \lambda_z\lambda_x E^\sigma(x)(z).\end{aligned}$$

Nota: $e^\sigma(x^\sigma) = e((x^\sigma)^{\sigma^{-1}})^\sigma = e(x)^\sigma$. □

Definição 4.4.9 (Funções auxiliares) Seja $\mu : \mathcal{Q} \to \mathbb{D}$ o único morfismo definido por $\eta : \mathcal{Q} \to F(\mathcal{Q})$ na coálgebra final. Definem-se as seguintes funções auxiliares:

$$\begin{aligned}par(d, d') &= \mu(d|d'), \\ new(e) &= \mu(e), \\ repl(d) &= \mu(!d).\end{aligned}$$

□

Lema 4.4.10 *As funções par, new e repl são morfismos de conjuntos-*Π.

Demonstração Vejamos o caso de *par*, atendendo a que μ é morfismo: $par((d,d')^\sigma) = par(d^\sigma, d'^\sigma) = \mu(d^\sigma|d'^\sigma) = \mu((d|d')^\sigma) = \mu(d|d')^\sigma = par(d|d')^\sigma$. Os outros casos demonstram-se de modo semelhante. □

Vamos seguidamente definir a semântica denotacional para o caso em que não estão envolvidas substituições e depois, à custa desta, definir a semântica para o caso geral. A cada operador sintáctico fazemos corresponder uma operação semântica sobre a semântica das componentes originais.

Definição 4.4.11 (Semântica denotacional fechada) A semântica denotacional fechada $\mathcal{D}_{fe} : \mathsf{Proc} \to \mathbb{D}$ é dada por

$$\begin{aligned}
\mathcal{D}_{fe}(\overline{a}\langle x\rangle.P) &= out(\overline{a}, x, \mathcal{D}_{fe}(P)),\\
\mathcal{D}_{fe}(a(x).P) &= in(a, \lambda_z\, \mathcal{D}_{fe}(P\{z/x\})),\\
\mathcal{D}_{fe}(\tau.P) &= sil(\mathcal{D}_{fe}(P)),\\
\mathcal{D}_{fe}(\mathsf{0}) &= null,\\
\mathcal{D}_{fe}(P+Q) &= sum(\mathcal{D}_{fe}(P), \mathcal{D}_{fe}(Q)),\\
\mathcal{D}_{fe}(P|Q) &= par(\mathcal{D}_{fe}(P), \mathcal{D}_{fe}(Q)),\\
\mathcal{D}_{fe}((\nu x)P) &= new(\lambda_z\, \mathcal{D}_{fe}(P\{z/x\})),\\
\mathcal{D}_{fe}(!P) &= repl(\mathcal{D}_{fe}(P)).
\end{aligned}$$

□

Para aliviar a escrita vamos muitas vezes denotar $\mathcal{D}_{fe}$ simplesmente por $\mathcal{D}$.

Lema 4.4.12 $\mathcal{D}_{fe}$ *é um morfismo de conjuntos-*Π.

Demonstração Vamos mostrar por indução na estrutura que para todo o $P \in \mathsf{Proc}$ tem-se $\mathcal{D}_{fe}(P^\sigma) = \mathcal{D}_{fe}(P)^\sigma$. O resultado é imediato para $P = \mathsf{0}$. Vejamos os restantes casos.

$$\begin{aligned}
\mathcal{D}((\tau.P)^\sigma) &= \mathcal{D}(\tau.P^\sigma) \\
&= sil(\mathcal{D}(P^\sigma)) \\
&= sil(\mathcal{D}(P)^\sigma) \qquad \text{(por hipótese de indução)} \\
&= \chi^{-1}(\mathcal{D}(P)^\sigma) \\
&= \chi^{-1}(\mathcal{D}(P))^\sigma \qquad \text{(porque } \chi \text{ é morfismo de conjuntos-}\Pi) \\
&= \mathcal{D}(\tau.P)^\sigma.
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}((a(x).P)^\sigma) &= \mathcal{D}((a^\sigma(x^\sigma).P^\sigma) \\
&= in(a^\sigma, \lambda_z \mathcal{D}(P^\sigma\{z/x^\sigma\}) \\
&= in(a^\sigma, \lambda_z \mathcal{D}((P\{z^{\sigma^{-1}}/x\})^\sigma) \\
&= in(a^\sigma, \lambda_z \mathcal{D}(P\{z^{\sigma^{-1}}/x\})^\sigma) \qquad \text{(por hipótese de indução)} \\
&= in(a^\sigma, (\lambda_z \mathcal{D}(P\{z/x\}))^\sigma) \\
&= \chi^{-1}((a^\sigma, (\lambda_z \mathcal{D}(P\{z/x\}))^\sigma)) \\
&= \chi^{-1}((a, \lambda_z \mathcal{D}(P\{z/x\})))^\sigma \\
&= in(a, \lambda_z \mathcal{D}(P\{z/x\})^\sigma \\
&= \mathcal{D}(a(x).P)^\sigma.
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}((P|Q)^\sigma) &= \mathcal{D}(P^\sigma|Q^\sigma) \\
&= par(\mathcal{D}(P^\sigma), \mathcal{D}(Q^\sigma) \\
&= par(\mathcal{D}(P)^\sigma, \mathcal{D}(Q)^\sigma) \qquad \text{(por hipótese de indução)} \\
&= par((\mathcal{D}(P), \mathcal{D}(Q))^\sigma) \\
&= par(\mathcal{D}(P), \mathcal{D}(Q))^\sigma \qquad \text{(porque } \textit{par} \text{ é morfismo)} \\
&= \mathcal{D}(P|Q)^\sigma.
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}(((\nu x)P)^\sigma) &= \mathcal{D}((\nu x^\sigma)P^\sigma) \\
&= new(\lambda_z \mathcal{D}(P^\sigma\{z/x^\sigma\})) \\
&= new(\lambda_z \mathcal{D}((P\{z^{\sigma^{-1}}/x\})^\sigma)) \\
&= new(\lambda_z \mathcal{D}(P\{z^{\sigma^{-1}}/x\})^\sigma) \qquad \text{(por hipótese de indução)} \\
&= new((\lambda_z \mathcal{D}(P\{z/x\}))^\sigma) \\
&= new(\lambda_z \mathcal{D}(P\{z/x\}))^\sigma \qquad \text{(porque } \textit{new} \text{ é morfismo).} \\
&= \mathcal{D}((\nu x)P)^\sigma
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}((!P)^\sigma) &= \mathcal{D}(!P^\sigma) \\
&= repl(\mathcal{D}(P^\sigma)) \\
&= repl(\mathcal{D}(P)^\sigma) && \text{(por hipótese de indução)} \\
&= repl(\mathcal{D}(P))^\sigma && \text{(porque } repl \text{ é morfismo)} \\
&= \mathcal{D}(!P)^\sigma.
\end{aligned}$$

Os casos em que o processo tem a forma $\overline{a}\langle x\rangle.P$ e $P+Q$ demonstram-se de modo semelhante aos casos $a(x).P$ e $P|Q$, respectivamente. □

Vamos em seguida definir a semântica denotacional no caso geral.

Definição 4.4.13 (Semântica denotacional) A semântica denotacional aberta $\mathcal{D}_{ab} : \mathsf{Proc} \to Env \to \mathbb{D}$ é dada por

$$\mathcal{D}_{ab}(P)\sigma = \mathcal{D}_{fe}(P\sigma).$$

□

4.5 Equivalência entre as Semânticas Operacional e Denotacional

Nesta secção vamos mostrar que a semântica operacional e a semântica denotacional, que foram definidas nas secções anteriores para o cálculo-π são equivalentes no caso fechado. A demonstração da equivalência será feita mostrando que a semântica operacional é um morfismo da coálgebra (Proc, γ) na coálgebra final $(\mathbb{D}, \chi)$ porque, pela unicidade do morfismo, como $\mathcal{O}$ foi definida como o único morfismo nestas condições tem-se $\mathcal{O} = \mathcal{D}$ (no caso fechado).

Comecemos por apresentar algumas definições e resultados auxiliares.

Lema 4.5.1 *Para todo o* $P \in \mathsf{Proc}$, $\mu(\mathcal{D}(P)) = \mathcal{D}(P)$

Demonstração Consideremos a função inclusão $inc : \mathbb{D} \to \mathbb{D} \subseteq \mathcal{Q}$. Esta função é um morfismo e $\mu \circ inc : \mathbb{D} \to \mathbb{D}$ é um morfismo na coálgebra final pelo que se tem $\mu \circ inc = \mathbf{1}_{\mathbb{D}}$. Desse modo, para todo o $p \in D$, $\mu(p) = \mu(inc(p)) = \mathbf{1}_{\mathbb{D}}(p) = p$, o que demonstra o resultado visto que $\mathcal{D}(P) \in \mathbb{D}$. □

Consideremos o functor F_0 que descreve o tipo das transições do sistema. Dado um morfismo de conjuntos nominais $f : Q \to R$ tem-se $F_0(f) : F_0(Q) \to F_0(R)$ e, para todo o $\delta \in F_0(Q)$, vamos usar a seguinte notação

$$f^*(\delta) = F_0(f)(\delta).$$

Lema 4.5.2 *Para todo o* $e(x), t(x) \in \mathcal{Q}_n, n \geq 0$,

$$\mu(\lambda_x \mu(e(x)|t(x))) = \mu(\lambda_x(e(x)|t(x)))$$

Demonstração Como se tratam de elementos da coálgebra final, para garantir a igualdade, basta-nos provar que $\mu(\lambda_x \mu(e(x)|t(x)))$ e $\mu(\lambda_x(e(x)|t(x)))$ são bissimilares. Para isso vamos verificar que a relação

$$R = \{(\mu(\lambda_x \mu(E(x))),\ \mu(\lambda_x E(x))) : \exists_{n \geq 0} \text{ tal que } E \in (\mathcal{Q}_n \times \mathcal{Q}_n)^{\mathcal{N}}\}$$

$$\cup \{(p, p) : p \in \mathbb{D}\}$$

é uma bissimulação.

1. $\mu(\lambda_x \mu(E(x))) \xrightarrow{\tau}_{\chi} \epsilon \Rightarrow \exists_{\epsilon'}\ \mu(\lambda_x E(x)) \xrightarrow{\tau}_{\chi} \epsilon'\ \&\ (\epsilon,\ \epsilon') \in R.$

Se $\mu(\lambda_x \mu(E(x))) \xrightarrow{\tau}_{\chi} \epsilon$ então existe e tal que $\lambda_x \mu(E(x)) \xrightarrow{\tau}_{\eta} e\ \&\ \mu^*(e) = \epsilon$ e por res_τ tem-se
$\mu(E(x)) \xrightarrow{\tau}_{\eta} e(x)$ para todo o $x \notin sup(\lambda_x \mu(E(x)))$, escolha-se x tal que $x \notin sup(\lambda_x E(x))$.
Então $\mu(E(x)) \xrightarrow{\tau}_{\chi} e(x)$
$\Rightarrow \exists_{E'(x)}\ E(x) \xrightarrow{\tau}_{\eta} E'(x)\ \&\ \mu^*(E'(x)) = e(x)$
e como $sup(\mu(E(x))) \subseteq sup(E(x))$, por res_τ

$\Rightarrow \lambda_x E(x) \xrightarrow{\tau}_\eta \lambda_x E'(x)$
$\Rightarrow \mu(\lambda_x E(x)) \xrightarrow{\tau}_\chi \mu^*(\lambda_x E'(x))$.
$\Rightarrow \mu(\lambda_x E(x)) \xrightarrow{\tau}_\chi \mu(\lambda_x E'(x))$.
Nestas condições $\epsilon = \mu(e) = \mu(\lambda_x e(x)) = \mu(\lambda_x \mu(E'(x)))$
e logo $(\mu(\lambda_x \mu(E'(x))),\ \mu(\lambda_x E'(x))) \in R$.

2. $\mu(\lambda_x \mu(E(x))) \xrightarrow{\overline{a}}_\chi \langle x \rangle \epsilon \Rightarrow \exists_{\epsilon'}\, \mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_\chi \langle x \rangle \epsilon' \,\&\, (\epsilon,\, \epsilon') \in R$.

A demonstração é semelhante à do caso anterior.

3. $\mu(\lambda_x \mu(E(x))) \xrightarrow{\overline{a}}_\chi e \Rightarrow \exists_{e'} \mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_\chi e'$

$$\&\, \forall_x\, (e(x), e'(x)) \in R$$

Se $\mu(\lambda_x \mu(E(x))) \xrightarrow{\overline{a}}_\chi e$ então existe δ tal que $\lambda_x E(x) \xrightarrow{\overline{a}}_\eta \delta \,\&\, \mu^*(\delta) = e$. Existem três possibilidades. O caso em que a transição é obtida por res_{out} é análogo aos casos anteriores e o caso em que é obtida por $res_{bo\,in}$ é semelhante ao caso 4. Vamos analisar apenas o caso em que a transição é obtida por res_{bo}.
Se $\lambda_x \mu(E(x)) \xrightarrow{\overline{a}}_\eta \lambda_x e''(x)$ com
$\mu(E(x)) \xrightarrow{\overline{a}}_\eta \langle x \rangle e''(x)$ para $x \notin sup(\lambda_x \mu(E(x)))$, $x \neq a$ vem
$\Rightarrow \exists_{E'(x)}\, E(x) \xrightarrow{\overline{a}}_\eta \langle x \rangle E'(x) \,\&\, \mu^*(\langle x \rangle E'(x)) = \langle x \rangle e''(x)$
e como $sup(\mu(E(x))) \subseteq sup(E(x))$, $x \neq a$, por res_{bo}
$\Rightarrow \lambda_x E(x) \xrightarrow{\overline{a}}_\eta \lambda_x E'(x)$
$\Rightarrow \mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_\chi \mu^*(\lambda_x E'(x))$.
$\Rightarrow \mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_\chi \lambda_x \mu(E'(x))$.
Nestas condições $e = \mu^*(\lambda_x e''(x)) = \lambda_x \mu(e''(x)) = \lambda_x \mu(\mu(E'(x))) = \lambda_x \mu(E'(x))$
e logo para todo o x tem-se $(\mu(E'(x)), \mu(E'(x))) \in R$.

4. $\mu(\lambda_x \mu(E(x))) \xrightarrow{a}_\chi t \Rightarrow \exists_{t'} \mu(\lambda_x E(x)) \xrightarrow{a}_\chi t' \,\&\, \forall_z\, (t(z), t'(z)) \in R$.

Se $\mu(\lambda_x \mu(E(x))) \xrightarrow{a}_\chi t$ então existe e tal que $\lambda_x \mu(E(x)) \xrightarrow{a}_\eta e \,\&\, \mu^*(e) = t$ e por $res_{bo\,in}$ terá de ser $e = \lambda_z \lambda_x T(x)(z)$ com

$\mu(E(x)) \xrightarrow{a}_{\eta} T(x)$ para $x \notin sup(\lambda_x\mu(E(x))) \cup sup(T),\ x \neq a.$
$\Rightarrow \mu(E(x)) \xrightarrow{a}_{\chi} T(x)$
$\Rightarrow \exists_{E'(x)}\ E(x) \xrightarrow{a}_{\eta} E'(x)\ \&\ \mu^*(E'(x)) = T(x)$
e como $sup(\mu(E(x))) \subseteq sup(E(x))\ \&\ x \neq a$, por $res_{bo\,in}$
$\Rightarrow \lambda_x E(x) \xrightarrow{a}_{\eta} \lambda_z\lambda_x E'(x)(z)$
$\Rightarrow \mu(\lambda_x E(x)) \xrightarrow{a}_{\chi} \mu^*(\lambda_z\lambda_x E'(x)(z))$
$\Rightarrow \mu(\lambda_x E(x)) \xrightarrow{a}_{\chi} \lambda_z\mu(\lambda_x E'(x)(z)).$
Nestas condições $t = \mu^*(\lambda_z\lambda_x T(x)(z)) = \lambda_z\mu(\lambda_x T(x)(z)) = \lambda_z\mu(\lambda_x\mu(E'(x)(z)))$,
porque $\mu^*(E'(x)) = T(x) \Rightarrow \mu(E'(x)) = T(x) \Rightarrow \mu(E'(x))(z) = T(x)(z) \Rightarrow$
$\mu(E'(x)(z)) = T(x)(z)$, e logo para todo o z,
$(\mu(\lambda_x\mu(E'(x)(z))),\ \mu(\lambda_x E'(x)(z))) \in R.$

Vejamos agora o caso da relação simétrica.

1. $\mu(\lambda_x E(x)) \xrightarrow{\tau}_{\chi} \epsilon \Rightarrow \exists_{\epsilon'}\mu(\lambda_x\mu(E(x))) \xrightarrow{\tau}_{\chi} \epsilon'\ \&\ (\epsilon,\ \epsilon') \in R$

Se $\mu(\lambda_x E(x)) \xrightarrow{\tau}_{\chi} \epsilon$ então existe E' tal que $\lambda_x E(x) \xrightarrow{\tau}_{\eta} \lambda_x E'(x)$
$\&\ \mu^*(\lambda_x E'(x)) = \epsilon$, com (por res_τ) $E(x) \xrightarrow{\tau}_{\eta} E'(x)$ para $x \notin sup(\lambda_x E(x))$.
O que implica $\mu(E(x)) \xrightarrow{\tau}_{\chi} \mu(E'(x))$ e, novamente por res_τ como
$sup(\lambda_x\mu(E(x))) \subseteq sup(\lambda_x E(x))$, vem
$\lambda_x\mu(E(x)) \xrightarrow{\tau}_{\eta} \lambda_x\mu(E'(x))$
$\Rightarrow \mu(\lambda_x\mu(E(x))) \xrightarrow{\tau}_{\chi} \mu^*(\lambda_x\mu(E'(x)))$
$\Rightarrow \mu(\lambda_x\mu(E(x))) \xrightarrow{\tau}_{\chi} \mu(\lambda_x\mu(E'(x))).$
Nestas condições $\epsilon = \mu^*(\lambda_x E'(x)) = \mu(\lambda_x E'(x))$
e logo $(\mu(\lambda_x E'(x)),\ \mu(\lambda_x\mu(E'(x)))) \in R.$

2. $\mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_{\chi} \langle x\rangle\epsilon \Rightarrow \exists_{\epsilon'}\mu(\lambda_x\mu(E(x))) \xrightarrow{\overline{a}}_{\chi} \langle x\rangle\epsilon'\ \&\ (\epsilon,\ \epsilon') \in R$

A demonstração é semelhante à do caso anterior.

3. $\mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_{\chi} e \Rightarrow \exists_{e'}\mu(\lambda_x\mu(E(x))) \xrightarrow{\overline{a}}_{\chi} e'$

$\&\ \forall_x\ (e(x), e'(x)) \in R$

Se $\mu(\lambda_x E(x)) \xrightarrow{\overline{a}}_\chi e$ então existe δ tal que $\lambda_x E(x) \xrightarrow{\overline{a}}_\eta \delta \,\&\, \mu^*(\delta) = e$. Pela definição de η existem três possibilidades, vamos analisar apenas o caso em que a transição é obtida por res_{bo}.
Se $\lambda_x E(x) \xrightarrow{\overline{a}}_\eta \lambda_x E'(x)$ com $E(x) \xrightarrow{\overline{a}}_\eta \langle x\rangle E'(x)$ para $x \notin sup(E),\ x \neq a$ vem
$\Rightarrow \mu(E(x)) \xrightarrow{\overline{a}}_\chi \mu^*(\langle x\rangle E'(x))$
$\Rightarrow \mu(E(x)) \xrightarrow{\overline{a}}_\chi \langle x\rangle \mu(E'(x))$
e como $sup(\mu(E(x))) \subseteq sup(E(x)),\ x \neq a$, por res_{bo}
$\Rightarrow \lambda_x \mu(E(x)) \xrightarrow{\overline{a}}_\eta \lambda_x \mu(E'(x))$
$\Rightarrow \mu(\lambda_x \mu(E(x))) \xrightarrow{\overline{a}}_\chi \mu^*(\lambda_x \mu(E'(x)))$
$\Rightarrow \mu(\lambda_x \mu(E(x))) \xrightarrow{\overline{a}}_\chi \lambda_x \mu^(\mu(E'(x)))$
$\Rightarrow \mu(\lambda_x \mu(E(x)) \xrightarrow{\overline{a}}_\chi \lambda_x \mu(E'(x))$
Nestas condições $e = \mu^*(\lambda_x E'(x)) = \lambda_x \mu(E'(x))$
e logo para todo o x tem-se $(\mu(E'(x)), \mu(E'(x))) \in R$.

4. $\mu(\lambda_x E(x)) \xrightarrow{a}_\chi t \Rightarrow \exists_{t'} \mu(\lambda_x \mu(E(x))) \xrightarrow{a}_\chi t' \,\&\, \forall_z\, (t(z), t'(z)) \in R$

Se $\mu(\lambda_x E(x)) \xrightarrow{a}_\chi t$ então existe e tal que $\lambda_x E(x) \xrightarrow{a}_\eta e \,\&\, \mu^*(e) = t$ e por $res_{bo\,in}$ terá de ser $e = \lambda_z \lambda_x T(x)(z)$ com
$E(x) \xrightarrow{a}_\eta T(x)$ para $x \notin sup(\lambda_x E(x)) \cup sup(T),\ x \neq a$.
$\Rightarrow \mu(E(x)) \xrightarrow{a}_\chi \mu(T(x))$
e como $sup(\mu(E(x))) \subseteq sup(E(x)) \,\&\, x \neq a$, por $res_{bo\,in}$
$\Rightarrow \lambda_x \mu(E(x)) \xrightarrow{a}_\eta \lambda_z \lambda_x \mu(T(x)(z)) \,\&\, \mu^*(E'(x)) = T(x)$
$\Rightarrow \mu(\lambda_x \mu(E(x))) \xrightarrow{a}_\eta \mu^*(\lambda_z \lambda_x \mu(T(x)(z)))$
$\Rightarrow \mu(\lambda_x \mu(E(x))) \xrightarrow{a}_\eta \lambda_z \mu(\lambda_x \mu(T(x)(z)))$
Nestas condições $t = \mu^*(\lambda_z \lambda_x T(x)(z)) = \lambda_z \mu(\lambda_x T(x)(z))$ e logo para todo o z, $(\mu(\lambda_x E'(x)(z)),\ \mu(\lambda_x \mu(E'(x)(z)))) \in R$. □

Estamos agora em condições de apresentar o resultado central desta secção que é estabelecer a equivalência entre a semântica operacional e a semântica denotacional no caso fechado.

Teorema 4.5.3 *Para todo o processo P, a semântica denotacional fechada de P é igual à semântica operacional de P, isto é*

$$\mathcal{D}_{fe}(P) = \mathcal{O}(P).$$

Demonstração Como $\mathcal{O}$ é o único morfismo de Proc em $\mathbb{D}$, se mostrarmos que $\chi \circ \mathcal{D}_{fe} = F(\mathcal{D}_{fe}) \circ \gamma$ tem-se necessariamente $\mathcal{D}_{fe}(P) = \mathcal{O}(P)$.
Se $P = \mathsf{0}$ vem $\chi(\mathcal{D}(\mathsf{0})) = \chi(\emptyset_n) = \emptyset = F(\mathcal{D})(\emptyset) = F(\mathcal{D})(\gamma(\mathsf{0})) = F(\mathcal{D}) \circ \gamma(\mathsf{0})$.
Se o processo tiver a forma $\overline{a}\langle x\rangle.P$ vem

$$\begin{aligned}
\chi(\mathcal{D}(\overline{a}\langle x\rangle.P)) &= \chi(out(\overline{a}, x, \mathcal{D}(P)) \\
&= \chi(\chi^{-1}(\{(\overline{a}, x, \mathcal{D}(P))\})) \\
&= \{(\overline{a}, x, \mathcal{D}(P))\} \\
&= F(\mathcal{D})(\{(\overline{a}, x, P)\} \\
&= F(\mathcal{D})(\gamma(\overline{a}\langle x\rangle.P)).
\end{aligned}$$

Se o processo tiver a forma $a(x).P$ vem

$$\begin{aligned}
\chi(\mathcal{D}(a(x).P)) &= \chi(in(a, \lambda_z \mathcal{D}(P\{z/x\})) \\
&= \chi(\chi^{-1}(\{(a, \lambda_z \mathcal{D}(P\{z/x\}))\})) \\
&= \{(a, \lambda_z \mathcal{D}(P\{z/x\}))\} \\
&= F(\mathcal{D})(\{(a, \lambda_z P\{z/x\})\} \\
&= F(\mathcal{D})(\gamma(a(x).P)).
\end{aligned}$$

O caso $\tau.P$ é idêntico aos anteriores.
Para $P + Q$ vem

$$\begin{aligned}
\chi(\mathcal{D}(P+Q)) &= \chi(sum(\mathcal{D}(P), \mathcal{D}(Q)) \\
&= \chi(\chi^{-1}(\chi(\mathcal{D}(P)) \cup \chi(\mathcal{D}(Q)))) \\
&= \chi(\mathcal{D}(P)) \cup \chi(\mathcal{D}(Q)) \\
&= F(\mathcal{D})(\gamma(P)) \cup F(\mathcal{D})(\gamma(Q)) \ \text{(por hipótese de indução)} \\
&= F(\{(\alpha, \delta) : P \xrightarrow{\alpha} \delta\}) \cup F(\{(\alpha, \delta) : Q \xrightarrow{\alpha} \delta\}) \\
&= F(\{(\alpha, \delta) : P \xrightarrow{\alpha} \delta\} \cup \{(\alpha, \delta) : P \xrightarrow{\alpha} \delta\}) \\
&= F(\gamma(P+Q)).
\end{aligned}$$

Para os restantes casos, mostrar que $\chi \circ \mathcal{D} = F(\mathcal{D}) \circ \gamma$ é equivalente a mostrarmos as seguintes duas implicações:

$$P \xrightarrow{\alpha}_\gamma \delta \Rightarrow \mathcal{D}_{fe}(P) \xrightarrow{\alpha}_\chi \mathcal{D}^*(\delta);$$
$$\mathcal{D}_{fe}(P) \xrightarrow{\alpha}_\chi \epsilon \Rightarrow \exists_\delta,\ P \xrightarrow{\alpha}_\gamma \delta \,\&\, \mathcal{D}^*(\delta) = \epsilon.$$

Comecemos pela primeira implicação.

1. Caso o processo tenha a forma $(\nu x)P$ tem-se

   (ν_τ) Se $(\nu x)P \xrightarrow{\tau}_\gamma (\nu x)P'$, pela definição de γ regra ν_τ, tem-se $P \xrightarrow{\tau}_\gamma P'$ e, por hipótese de indução na estrutura do processo, obtém-se $\mathcal{D}(P) \xrightarrow{\tau}_\chi \mathcal{D}(P')$
   donde, pela definição de η,
   $\Rightarrow \mathcal{D}(P) \xrightarrow{\tau}_\eta \mathcal{D}(P')$
   $\Rightarrow \lambda_z\mathcal{D}(P\{z/x\}) \xrightarrow{\tau}_\eta \lambda_z\mathcal{D}(P'\{z/x\})$ (por res_τ)
   $\Rightarrow \mu(\lambda_z\mathcal{D}(P\{z/x\})) \xrightarrow{\tau}_\chi \mu^*(\lambda_z\mathcal{D}(P'\{z/x\}))$
   $\Rightarrow \mu(\lambda_z\mathcal{D}(P\{z/x\})) \xrightarrow{\tau}_\chi \mu^(\lambda_z\mathcal{D}(P'\{z/x\}))$
   o que é equivalente a $\mathcal{D}((\nu x)P) \xrightarrow{\tau}_\chi \mathcal{D}((\nu x)P')$ como se pretendia.

   (ν_{out}) Se $(\nu x)P \xrightarrow{\overline{a}}_\gamma \langle y\rangle(\nu x)P'$ com $P \xrightarrow{\overline{a}}_\gamma \langle y\rangle P'$ para $x \neq a, x \neq y$ a demonstração é semelhante à anterior.

   (ν_{bo}) Se $(\nu x)P \xrightarrow{\overline{a}}_\gamma \lambda_z P'\{z/x\}$ com $P \xrightarrow{\overline{a}}_\gamma \langle x\rangle P'$ para $x \neq a$, por hipótese de indução na estrutura do processo
   $\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \langle x\rangle\mathcal{D}(P')$ e, pela definição de η,
   $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \langle x\rangle\mathcal{D}(P')$ e como $x \notin sup(\lambda_z\mathcal{D}(P\{z/x\}))$ por res_{out}
   $\lambda_z\mathcal{D}(P\{z/x\}) \xrightarrow{\overline{a}}_\eta \lambda_z\mathcal{D}(P'\{z/x\})$
   $\Rightarrow \mu(\lambda_z\mathcal{D}(P\{z/x\})) \xrightarrow{\overline{a}}_\chi \mu^*(\lambda_z\mathcal{D}(P'\{z/x\}))$
   $\Rightarrow \mu(\lambda_z\mathcal{D}(P\{z/x\})) \xrightarrow{\overline{a}}_\chi \lambda_z\mu(\mathcal{D}(P'\{z/x\}))$
   Pelo lema 4.5.1 para todo o z tem-se $\mu(\mathcal{D}(P'\{z/x\})) = \mathcal{D}(P'\{z/x\})$
   logo
   $\Rightarrow \mu(\lambda_z\mathcal{D}(P\{z/x\})) \xrightarrow{\overline{a}}_\chi \lambda_z\mathcal{D}(P'\{z/x\})$
   $\Rightarrow \mathcal{D}((\nu x)P) \xrightarrow{\overline{a}}_\chi \mathcal{D}^*(\lambda_z P'\{z/x\})$.

($\nu_{bo\,in}$) Se $(\nu x)P \xrightarrow{a}_{\gamma} \lambda_z((\nu x)Q)\{z/w\}$ com $P \xrightarrow{a}_{\gamma} \lambda_z P'\{z/w\}$ para $x \neq a, w$, por hipótese de indução $\mathcal{D}(P) \xrightarrow{a}_{\chi} \mathcal{D}^*(\lambda_z Q\{z/w\})$ o que implica $\mathcal{D}(P) \xrightarrow{a}_{\eta} \mathcal{D}^*(\lambda_z Q\{z/w\})$. Seja
$e = \lambda_v \mathcal{D}(P\{v/x\})$;
$T = \lambda_v \lambda_z \begin{cases} \mathcal{D}(Q\{z/w\}\{v/x\}), & \text{se } z \neq x; \\ \mathcal{D}(Q\{y/x\}\{x/w\}\{v/y\}),\ y \notin lv(Q) & \text{se } z = x. \end{cases}$
Tem-se $T(x)(z) = \mathcal{D}(Q\{z/w\})$ para todo o $x, z \in \mathcal{N}$ e para todo o w a função tem suporte finito e a própria T tem suporte finito. Nestas condições $\mathcal{D}(P) = e(x)$ e $\lambda_z \mathcal{D}(Q\{z/w\}) = T(x)$ e logo

$$\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \lambda_z \lambda_v T(v)(z)$$

Caso $z \neq x$ tem-se
$\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \lambda_z \lambda_v \mathcal{D}(Q\{z/w\}\{v/x\})$
$\Rightarrow \mu(\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \mu^*(\lambda_z \lambda_v \mathcal{D}(Q\{z/w\}\{v/x\}))$
$\Rightarrow \mu(\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \lambda_z \mu(\lambda_v \mathcal{D}(Q\{z/w\}\{v/x\}))$
$\Rightarrow \mathcal{D}((\nu x)P) \xrightarrow{a}_{\chi} \lambda_z \mathcal{D}((\nu x)Q\{z/w\})$
Mas como $z \neq x$, $(\nu x)Q\{z/w\} = ((\nu x)Q)\{z/w\}$ e portanto
$\mathcal{D}((\nu x)P) \xrightarrow{a}_{\chi} \lambda_z \mathcal{D}(((\nu x)Q)\{z/w\})$
$\Rightarrow \mathcal{D}((\nu x)P) \xrightarrow{a}_{\chi} \mathcal{D}^*(\lambda_z((\nu x)Q)\{z/w\})$.
Caso $z = x$ vem
$\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \lambda_z \lambda_v \mathcal{D}(Q\{y/x\}\{x/w\}\{v/y\})$, para $y \notin lv(Q)$
$\Rightarrow \mu(\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \mu^*(\lambda_z \lambda_v \mathcal{D}(Q\{y/x\}\{x/w\}\{v/y\}))$
$\Rightarrow \mu(\lambda_v \mathcal{D}(P\{v/x\}) \xrightarrow{a}_{\eta} \lambda_z \mu(\lambda_v \mathcal{D}(Q\{y/x\}\{x/w\}\{v/y\}))$
$\Rightarrow \mathcal{D}((\nu x)P) \xrightarrow{a}_{\chi} \lambda_z \mathcal{D}((\nu y)Q\{y/x\}\{x/w\})$
Mas como $y \notin lv(Q)$, $(\nu y)Q\{y/x\}\{x/w\} = ((\nu x)Q)\{x/w\}$ pelo que a transição é equivalente a
$\mathcal{D}((\nu x)P) \xrightarrow{a}_{\chi} \lambda_z \mathcal{D}(((\nu x)Q)\{x/w\})$ e atendendo a que $x = z$
$\Rightarrow \mathcal{D}((\nu x)P) \xrightarrow{a}_{\chi} \mathcal{D}^*(\lambda_z((\nu x)Q)\{z/w\})$.

Se a transição inicial fosse com $\overline{a}$ em vez de a a demonstração era semelhante.

2. Caso o processo tenha a forma $P|Q$ tem-se:

   ($comp_{out}$) Se $P|Q \xrightarrow{\tau}_\gamma \langle x\rangle(P'|Q)$ com $P \xrightarrow{\tau}_\gamma \langle x\rangle P'$, por hipótese de indução $\mathcal{D}(P) \xrightarrow{\tau}_\chi \langle x\rangle\mathcal{D}(P')$ donde, pela definição de η,
   $\mathcal{D}(P) \xrightarrow{\tau}_\eta \langle x\rangle\mathcal{D}(P')$
   $\Rightarrow \mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\tau}_\eta \langle x\rangle(\mathcal{D}(P')|\mathcal{D}(Q))$ (por par_{out})
   $\Rightarrow \mu(\mathcal{D}(P)|\mathcal{D}(Q)) \xrightarrow{\tau}_\chi \langle x\rangle\mu(\mathcal{D}(P')|\mathcal{D}(Q))$ o que, pela definição de $\mathcal{D}$ dá
   $\mathcal{D}(P|Q) \xrightarrow{\tau}_\chi \langle x\rangle\mathcal{D}(P'|Q) \Leftrightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_\chi \mathcal{D}^*(\langle x\rangle P'|Q)$.

   ($comp_\tau$) Demonstra-se de modo análogo ao anterior.

   ($comp_{bo\,in}$) Se $P|Q \xrightarrow{\overline{a}}_\gamma \lambda_z(P'|Q)\{z/w\}$ com $P \xrightarrow{\overline{a}}_\gamma \lambda_z P'\{z/w\}$ para $w \notin lv(Q)$, por hipótese de indução $\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \lambda_z\mathcal{D}(P'\{z/w\})$ donde, pela definição de η,
   $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \lambda_z\mathcal{D}(P'\{z/w\})$
   $\Rightarrow \mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\overline{a}}_\eta \lambda_z(\mathcal{D}(P'\{z/w\})|\mathcal{D}(Q))$ o que, por $par_{bo\,in}$, implica
   $\mu(\mathcal{D}(P)|\mathcal{D}(Q)) \xrightarrow{\overline{a}}_\chi \lambda_z\mu(\mathcal{D}(P'\{z/w\})|\mathcal{D}(Q))$, e atendendo à definição de $\mathcal{D}$ vem
   $\mathcal{D}(P|Q) \xrightarrow{\overline{a}}_\chi \lambda_z\mathcal{D}(P'\{z/w\}|Q)$
   $\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\overline{a}}_\chi \mathcal{D}^*(\lambda_z(P'\{z/w\}|Q))$
   $\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\overline{a}}_\chi \mathcal{D}^*(\lambda_z(P'|Q)\{z/w\})$, visto que $w \notin lv(Q)$.
   O caso em que a transição inicial é com a demonstra-se do mesmo modo.

   ($sync$) Se $P|Q \xrightarrow{\tau}_\gamma P'|Q'\{x/w\}$ com $P \xrightarrow{\overline{a}}_\gamma \langle x\rangle P' \,\&\, Q \xrightarrow{a} \lambda_z Q'\{z/w\}$, por hipótese de indução $\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \langle x\rangle\mathcal{D}(P') \,\&\, \mathcal{D}(Q) \xrightarrow{a}_\chi \lambda_z\mathcal{D}(Q'\{z/w\})$ donde, pela definição de η,
   $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \langle x\rangle\mathcal{D}(P') \,\&\, \mathcal{D}(Q) \xrightarrow{a}_\eta \lambda_z\mathcal{D}(Q'\{z/w\})$, donde por com
   $\mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\tau}_\eta \mathcal{D}(P')|\mathcal{D}(Q'\{x/w\}))$ o que implica
   $\mu(\mathcal{D}(P)|\mathcal{D}(Q)) \xrightarrow{\tau}_\chi \mu(\mathcal{D}(P')|\mathcal{D}(Q'\{x/w\}))$
   $\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_\chi \mathcal{D}(P'|Q'\{x/w\})$

$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mathcal{D}^*(P'|Q'\{x/w\})$

($sync_\nu$) Se $P|Q \xrightarrow{\tau}_{\gamma} (\nu x)(P'\{x/w\}|Q'\{x/w\})$ com $P \xrightarrow{\overline{a}}_{\gamma} \lambda_z P'\{z/w\}$ & $Q \xrightarrow{a}_{\gamma} \lambda_z Q'\{z/w\}$ & $x \notin lv(P') - \{w\} \cup lv(Q') - \{w\}$ e por hipótese de indução $\mathcal{D}(P) \xrightarrow{\overline{a}}_{\chi} \lambda_z \mathcal{D}(P'\{z/w\})$ & $\mathcal{D}(Q) \xrightarrow{a}_{\chi} \lambda_z \mathcal{D}(Q'\{z/w\})$ donde, pela definição de η,
$\mathcal{D}(P) \xrightarrow{\overline{a}}_{\eta} \lambda_z \mathcal{D}(P'\{z/w\})$ & $\mathcal{D}(Q) \xrightarrow{a}_{\eta} \lambda_z \mathcal{D}(Q'\{z/w\})$
$\Rightarrow \mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\tau}_{\eta} \lambda_z(\mathcal{D}(P'\{z/w\})|\mathcal{D}(Q'\{z/w\}))$ o que implica
$\mu(\mathcal{D}(P)|\mathcal{D}(Q)) \xrightarrow{\tau}_{\chi} \mu(\lambda_z(\mathcal{D}(P'\{z/w\})|\mathcal{D}(Q'\{z/w\})))$
$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mu(\lambda_z(\mathcal{D}(P'\{z/w\})|\mathcal{D}(Q'\{z/w\})))$
o que, atendendo ao lema 4.5.2, é equivalente a
$\mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mu(\lambda_z \mu(\mathcal{D}(P'\{z/w\})|\mathcal{D}(Q'\{z/w\})))$
$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mu(\lambda_z \mathcal{D}(P'\{z/w\}|Q'\{z/w\})))$
$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mu(\lambda_z(\mathcal{D}(P'\{x/w\}\{z/x\}|Q'\{x/w\}\{z/x\})))$
$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mu(\lambda_z(\mathcal{D}((P'\{x/w\}|Q'\{x/w\})\{z/x\})))$
$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mathcal{D}((\nu x)(P'\{x/w\}|Q'\{x/w\}))$
$\Rightarrow \mathcal{D}(P|Q) \xrightarrow{\tau}_{\chi} \mathcal{D}^*((\nu x)(P'\{x/w\}|Q'\{x/w\}))$
Nota: Como $x \notin lv(P') - \{w\} \cup lv(Q') - \{w\}$ tem-se $P'\{x/w\}\{z/x\} = P'\{z/w, z/x\} = P'\{z/w\}$ e do mesmo modo $Q'\{x/w\}\{z/x\} = Q'\{z/w\}$.

Os restantes casos demonstram-se de modo análogo.

3. Se o processo tiver a forma $!P$ as transições são obtidas à custa da congruência $!P \equiv !P|P$ e derivam das transições de $P|Q$ já estudadas, conforme indicado nos comentários que se seguem à definição do sistema de transições (4.3.6).

Passemos agora à demonstração da segunda implicação.

1. Caso o processo tenha a forma $(\nu x)P$:
se $\mathcal{D}_{fe}((\nu x)P) \xrightarrow{\alpha}_{\chi} \epsilon$ pela definição de $\mathcal{D}$ tem-se

$\mu(\lambda_z \mathcal{D}_{fe}(P\{z/x\})) \xrightarrow{\alpha}_\chi \epsilon$, o que implica

$$\exists_{\delta'},\ \lambda_z \mathcal{D}_{fe}(P\{z/x\}) \xrightarrow{\alpha}_\eta \delta' \,\&\, \mu^*(\delta') = \epsilon$$

(a) (res_τ) Se $\alpha = \tau$, atendendo às regras que definem η, a transição toma a forma
$\lambda_z \mathcal{D}_{fe}(P\{z/x\}) \xrightarrow{\tau}_\eta e$ com
$\mathcal{D}(P\{x/x\}) \xrightarrow{\tau}_\eta e(x)$ para $x \notin sup(\lambda_z \mathcal{D}_{fe}(P\{z/x\}))$
Como $\mathcal{D}(P) \in \mathbb{D}$ a única possibilidade é a transição ter resultado da regra *base* e então $\mathcal{D}(P) \xrightarrow{\tau}_\chi e(x)$ e por hipótese de indução
$\exists_Q :\ P \xrightarrow{\tau}_\gamma Q \,\&\, \mathcal{D}^*(Q) = e(x)$
donde, por ν_τ, vem $(\nu_x)P \xrightarrow{\tau}_\gamma (\nu x)Q$.
Por outro lado, neste caso $\mu^*(e) = \epsilon \Leftrightarrow \mu(e) = \epsilon$ e
$\mathcal{D}^*(Q) = e(x) \Leftrightarrow \mathcal{D}(Q) = e(x)$, donde

$$\begin{aligned}
\mathcal{D}^*((\nu x)Q) &= \mathcal{D}((\nu x)Q) \\
&= \mu(\lambda_x \mathcal{D}(Q\{w/w\})) \\
&= \mu(\lambda_x \mathcal{D}(Q)) \\
&= \mu(\lambda_x e(x)) \\
&= \mu(e) \\
&= \epsilon, \quad \text{como se pretendia.}
\end{aligned}$$

(b) Se $\alpha = \overline{a}$, atendendo às regras que definem η, temos três casos a considerar

(res_{out}) $\lambda_z \mathcal{D}(P\{z/x\}) \xrightarrow{\overline{a}}_\eta \langle y \rangle \lambda_z e(z)$ com $\mathcal{D}(P\{x/x\}) \xrightarrow{\overline{a}}_\eta \langle y \rangle e(x)$ para $x \notin sup(\lambda_z \mathcal{D}_{fe}(P\{z/x\}))$, $x \neq a$, $x \neq y$, e neste caso $\epsilon = \langle y \rangle \mu(\lambda_z e(z))$. Atendendo à definição de η a transição resultou da regra *base* logo $\mathcal{D}(P\{x/x\}) \xrightarrow{\overline{a}}_\chi \langle y \rangle e(x)$ e por hipótese de indução na estrutura do processo vem
$\exists_{P'} P \xrightarrow{\overline{a}}_\gamma \langle y \rangle P' \,\&\, \mathcal{D}^*(\langle y \rangle P') = \langle y \rangle e(x)$ donde $\mathcal{D}(P') = e(x)$
e, pela definição de γ (ν_{out}), vem,
$(\nu x)P \xrightarrow{\overline{a}}_\gamma \langle y \rangle (\nu x)P'$.

Nestas condições, $\mathcal{D}^*(\langle y\rangle(\nu x)P') = \langle y\rangle\mathcal{D}((\nu x)P')$
$= \langle y\rangle\mu(\lambda_x\mathcal{D}(P'\{x/x\}))$
$= \langle y\rangle\mu(\lambda_x e(x))$
$= \langle y\rangle\mu(e)$
$= \mu^*(\langle y\rangle e)$
$= \epsilon$, como se pretendia.

(res_{bo}) $\lambda_z\mathcal{D}(P\{z/x\}) \xrightarrow{\overline{a}}_\eta \lambda_z e(z)$ com $\mathcal{D}_{fe}(P\{x/x\}) \xrightarrow{\overline{a}}_\eta \langle x\rangle e(x)$ para $x \notin sup(\lambda_z\mathcal{D}_{fe}(P\{z/x\}))$, $x \neq a$, pela definição de η, a transição resultou da regra base, logo
$\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \langle x\rangle e(x)$ e, por hipótese de indução,
$\exists_Q : P \xrightarrow{\overline{a}}_\gamma \langle x\rangle Q \,\&\, \mathcal{D}^*(\langle x\rangle Q) = \langle x\rangle e(x)$ donde, como $x \neq a$ por ν_{bo} vem
$(\nu x)P \xrightarrow{\overline{a}}_\gamma \lambda z Q\{z/x\}$.
Neste caso tem-se $\langle x\rangle\mathcal{D}(Q) = \langle x\rangle e(x)$ logo $\mathcal{D}(Q) = e(x)$.
Atendendo ao lema 4.5.1 vem

$$\begin{aligned}\mathcal{D}^*(\lambda_z Q\{z/x\}) &= \mathcal{D}^*(\lambda_x Q\{x/x\})\\ &= \lambda_x\mathcal{D}(Q)\\ &= \lambda_x\mu(\mathcal{D}(Q)) \qquad \text{(pelo lema 4.5.1)}\\ &= \lambda_x\mu(e(x))\\ &= \epsilon\end{aligned}$$

($res_{bo\,in}$) Se $\lambda_z\mathcal{D}(P\{z/x\}) \xrightarrow{\overline{a}}_\eta \lambda_z\lambda_x T(x)(z)$ com $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \lambda_z T(x)(z)$ para $x \notin sup(\lambda_z\mathcal{D}(P\{z/x\})) \cup sup(\lambda_z\lambda_x T(x)(z))$, pela definição de η terá de ser $\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \lambda_z T(x)(z)$ e por hipótese de indução na estrutura do processo

$$\exists_Q\, P \xrightarrow{\overline{a}}_\gamma \lambda_z Q\{z/w\} \,\&\, \mathcal{D}^*(\lambda_z Q\{z/w\}) = \lambda_z T(x)(z)$$

Assim $\lambda_z\mathcal{D}(Q\{w/z\}) = \lambda_z\lambda_x T(x)(z)$ logo para todo o z, $\mathcal{D}(Q\{w/z\}) = \lambda_x T(x)(z)$ e para todo x, z, $\mathcal{D}(Q\{w/z\}) = T(x)(z)$. Por $\nu_{bo\,in}$ obtém-se $(\nu x)P \xrightarrow{\overline{a}} \lambda_z((\nu x)Q)\{z/w\}$ para $x \neq a, x \neq w$. Falta apenas verificar que

$$\mathcal{D}^*(\lambda_z((\nu x)Q)\{z/w\}) = \mu^*(\lambda_z\lambda_x T(x)(z)),$$

ou seja que

$$\lambda_z\mathcal{D}((\nu x)Q)\{z/w\}) = \lambda_z\mu(\lambda_x\mathcal{D}(Q\{w/z\})).$$

Caso $x \neq z$ vem,

$$\begin{aligned}\lambda_z\mathcal{D}((\nu x)Q)\{z/w\}) &= \lambda_z\mathcal{D}((\nu x)Q\{z/w\}) \\ &= \lambda_z\mu(\lambda_v\mathcal{D}(Q\{z/w\}\{v/x\})) \\ &= \lambda_z\mu(\lambda_x\mathcal{D}(Q\{z/w\})).\end{aligned}$$

Caso $x = z$ tem-se, para $y \notin lv(Q)$,

$$\begin{aligned}\lambda_z\mathcal{D}((\nu x)Q)\{z/w\}) &= \lambda_z\mathcal{D}((\nu_y)Q\{y/x\}\{z/w\}) \\ &= \lambda_z\mu(\lambda_v(\mathcal{D}(Q\{y/x\}\{z/w\}\{v/y\}))) \\ &= \lambda_z\mu(\lambda_v(\mathcal{D}(Q\{v/x,\, z/w\}))) \\ &= \lambda_z\mu(\lambda_x(\mathcal{D}(Q\{z/w\}))).\end{aligned}$$

A justificação para a terceira igualdade é que

$$\{v/y\} \circ \{z/w\} \circ \{y/x\} = \{v/x,\, z/w,\, v/y\}$$

mas como $y \notin lv(Q)$ tem-se

$$Q\{v/x,\, z/w,\, v/y\} = Q\{v/x,\, z/w\}.$$

(c) Se $\alpha = a$ a demonstração é idêntica ao caso $res_{bo\,in}$.

2. Caso o processo tenha a forma $P|Q$, se $\mathcal{D}(P|Q) \xrightarrow{\alpha}_\chi \epsilon$ pela definição de $\mathcal{D}$ tem-se $\mu(\mathcal{D}(P)|\mathcal{D}(Q)) \xrightarrow{\alpha}_\chi \epsilon$, o que implica

$$\exists_{\delta'}\ \mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\alpha}_\eta \delta' \,\&\, \mu^*(\delta') = \epsilon.$$

(a) Se $\alpha = \tau$ atendendo à definição de η existem três possibilidades distintas:

(par_τ) Se $\mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\tau}_\eta q|\mathcal{D}(Q)$ com $\mathcal{D}(P) \xrightarrow{\tau}_\eta q$, pela definição de η terá de ser $\mathcal{D}(Q) \xrightarrow{\tau}_\chi q$, donde por hipótese de indução

$$\exists_R : P \xrightarrow{\tau}_\gamma R \,\&\, \mathcal{D}^*(R) = q$$

e por $comp_\tau$ vem $P|Q \xrightarrow{\tau}_\gamma R|Q$.

Por outro lado, neste caso $\epsilon = \mu(q|\mathcal{D}(Q))$ e logo

$$\begin{aligned}\mathcal{D}^*(R|Q) &= \mathcal{D}(R|Q)\\ &= \mu(\mathcal{D}(R)|\mathcal{D}(Q))\\ &= \mu(q|\mathcal{D}(Q))\\ &= \epsilon \text{ , como se pretendia.}\end{aligned}$$

(*com*) Se $\mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\tau}_\eta q|t(x)$ com $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \langle x\rangle q$ e $\mathcal{D}(Q) \xrightarrow{a}_\eta t$, pela definição de η terá de ser $\mathcal{D}(Q) \xrightarrow{\overline{a}}_\chi \langle x\rangle q \,\&\, \mathcal{D}(Q) \xrightarrow{a}_\chi t$, donde por hipótese de indução

$(\exists_R : P \xrightarrow{\overline{a}}_\gamma \langle x\rangle R \,\&\, \mathcal{D}^*(\langle x\rangle R) = \langle x\rangle q) \,\&$

$(\exists_S : Q \xrightarrow{a}_\gamma \lambda_z S\{z/w\} \,\&\, \mathcal{D}^*(\lambda_z S\{z/w\}) = t)$

e por *sync* vem $P|Q \xrightarrow{\tau}_\gamma R|S\{x/w\}$.

Por outro lado, neste caso $\epsilon = \mu(q|t(x))$, $\mathcal{D}(R) = q$ e $\mathcal{D}(\lambda_z S\{z/w\}) = t$ donde $\forall_x\ \mathcal{D}(S\{x/w\}) = t(x)$, logo

$$\begin{aligned}\mathcal{D}^*(R|S\{x/w\}) &= \mathcal{D}(R|S\{x/w\})\\ &= \mu(\mathcal{D}(R)|\mathcal{D}(S\{x/w\}))\\ &= \mu(q|t(x))\\ &= \epsilon \text{ , como se pretendia.}\end{aligned}$$

(com_{res}) Se $\mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\tau}_\eta \lambda_x(e(x)|t(x))$ com $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \lambda_z e(z)$ e $\mathcal{D}(Q) \xrightarrow{a}_\eta \lambda_x t(z)$, pela definição de η terá de ser $\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \lambda_z e(z)$ e $\mathcal{D}(Q) \xrightarrow{a}_\chi \lambda_z t(z)$, donde por hipótese de indução

$$(\exists_R : P \xrightarrow{\overline{a}}_\gamma \lambda_z R\{z/w\} \,\&\, \mathcal{D}^*(\lambda_z R\{z/w\}) = \lambda_z e(z)) \,\&$$

$$(\exists_S : Q \xrightarrow{a}_\gamma \lambda_z S\{z/w\} \,\&\, \mathcal{D}^*(\lambda_z S\{z/w\}) = \lambda_z t(z))$$

e por $sync_\nu$ vem $P|Q \xrightarrow{\tau}_\gamma (\nu x)(R\{x/w\}|S\{x/w\})$ para $x \notin lv(R) - \{w\} \cup lv(S) - \{w\}$.

Por outro lado, neste caso $\epsilon = \mu(\lambda_x(e(x)|t(x)))$ e $\lambda_z\mathcal{D}(S\{z/w\}) = \lambda_z e(z) \,\&\, \lambda_z\mathcal{D}(S\{z/w\}) = \lambda_z t(z)$, pelo que

$$\begin{aligned}\mathcal{D}^*((\nu x)(R\{x/w\}|S\{x/w\})) &= \mathcal{D}((\nu x)(R\{x/w\}|S\{x/w\}))\\ &= \mu(\lambda_z\mathcal{D}((R\{x/w\}|S\{x/w\})\{z/x\}))\\ &= \mu(\lambda_z(\mathcal{D}(R\{x/w\}\{z/x\}|S\{x/w\}\{z/x\})))\\ &= \mu(\lambda_z(\mathcal{D}(R\{z/w\}|S\{z/w\})))\\ &= \mu(\lambda_z\mu(\mathcal{D}(R\{z/w\})|\mathcal{D}(S\{z/w\})))\\ &= \mu(\lambda_z(\mathcal{D}(R\{z/w\})|\mathcal{D}(S\{z/w\})))\\ &= \mu(\lambda_z(e(z)|t(z)))\\ &= \epsilon \text{ , como se pretendia.}\end{aligned}$$

A sexta igualdade resulta do lema 4.5.2. Em relação às substituições, note-se que $\{z/x\}\circ\{x/w\} = \{z/w,\, z/x\}$, e como $x \notin lv(R) - \{w\} \cup lv(S) - \{w\}R$ tem-se $R\{z/w,\, z/x\} = R\{z/w\}$ e $S\{z/w,\, z/x\} = S\{z/w\}$.

(b) Se $\alpha = \overline{a}$, atendendo à definição de η existem duas possibilidades.

(par_{out}) Se $\mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\overline{a}}_\eta \langle x\rangle(q|\mathcal{D}(Q))$ com $\mathcal{D}(P) \xrightarrow{\overline{a}}_\eta \langle x\rangle q$, tem-se também $\mathcal{D}(P) \xrightarrow{\overline{a}}_\chi \langle x\rangle q$ donde por hipótese de indução $\exists_R P \xrightarrow{\overline{a}}_\gamma \langle x\rangle R \,\&\, \mathcal{D}^*(\langle x\rangle R) = \langle x\rangle q$ e, atendendo à definição de γ, regra $comp_{out}$, vem $P|Q \xrightarrow{\overline{a}}_\gamma \langle x\rangle(R|Q)$.
Neste caso $\epsilon = \mu^*(\langle x\rangle(q|\mathcal{D}(Q))) = \langle x\rangle\mu(q|\mathcal{D}(Q))$ e $\langle x\rangle\mathcal{D}(R) = \langle x\rangle q$ donde $\mathcal{D}(R) = q$, logo

$$\begin{aligned}\mathcal{D}^*(\langle x\rangle(R|Q)) &= \langle x\rangle\mathcal{D}(R|Q)\\ &= \langle x\rangle\mu(\mathcal{D}(R)|\mathcal{D}(Q))\\ &= \langle x\rangle\mu(q|\mathcal{D}(Q))\\ &= \epsilon.\end{aligned}$$

($par_{bo\,in}$) Se $\mathcal{D}(P)|\mathcal{D}(Q) \xrightarrow{\overline{a}}_\eta \lambda_x(e(x)|\mathcal{D}(Q))$ com $q \xrightarrow{\overline{a}}_\eta e$ tem-se pela definição de η, $q \xrightarrow{\overline{a}}_\chi e$, donde por hipótese de indução
$\exists_R P \xrightarrow{\overline{a}}_\gamma \lambda_x R\{w/x\} \,\&\, \mathcal{D}^*(\lambda_x R\{x/w\}) = \lambda_x e(x)$
e, atendendo à definição de γ, regra $comp_{bo\,in}$,
$P|Q \xrightarrow{\overline{a}}_\gamma \lambda_x(R\{x/w\}|Q)$.

Nestas condições,

$$\begin{aligned}\mathcal{D}^*(\lambda_x(R\{x/w\}|Q)) &= \lambda_x\mathcal{D}(R\{x/w\}|Q)\\ &= \lambda_x\mu(\mathcal{D}(R\{x/w\})|\mathcal{D}(Q))\\ &= \langle x\rangle p(e(x)|\mathcal{D}(Q))\\ &= \epsilon\end{aligned}$$

(c) Se $\alpha = a$ existe uma única possibilidade, que é obtida pela regra $par_{bo\,in}$, cuja demonstração é análoga à do último caso estudado.

3. Caso o processo tenha a forma $!P$, se $\mathcal{D}(!P) \xrightarrow{\alpha}_\chi \epsilon$, pela definição de $\mathcal{D}$ tem-se $repl(\mathcal{D}(P)) \xrightarrow{\alpha}_\chi \epsilon \Leftrightarrow \mu(!\mathcal{D}(P)) \xrightarrow{\alpha}_\chi \epsilon$ o que implica

$$\exists_q : !\mathcal{D}(P) \xrightarrow{\alpha}_\eta q \,\&\, \mu^*(q) = \epsilon$$

mas as transições de $!\mathcal{D}(P)$ são obtidas pela congruência $!\mathcal{D}(P) \equiv !\mathcal{D}(P)|\mathcal{D}(P)$ e derivam das transições de $\mathcal{D}(P)|\mathcal{D}(Q)$ já estudadas. □

4.6 Bissimilaridade e congruência forte

Quando temos uma noção de semântica, isso introduz automaticamente uma relação de equivalência entre processos e é conveniente que a equivalência semântica induza a mesma equivalência no domínio semântico, nesta secção vamos verificar que isto acontece no nosso caso.

Uma equivalência que tem em conta as acções internas é usualmente designada por equivalência forte, e uma que não tem em contas estas acções designa-se por equivalência fraca. É vantajoso tratar primeiro as equivalências fortes ([SW01]) por ser um o caso mais fácil e porque depois pode ser útil para a equivalência fraca uma vez que a versão forte implica a versão fraca da equivalência. Nesta secção vamos fazer o estudo da bissimilaridade e congruência apenas no caso forte uma vez que não foi nosso objectivo explorar as equivalências em toda a sua amplitude.

O núcleo de equivalência de uma semântica define uma relação de equivalência que é designada por *relação de equivalência semântica*. O que vamos verificar nesta secção é que relação de equivalência semântica forte coincide com a relação de bissimilaridade das coálgebras-F, no caso fechado; e com a relação de congruência forte, no caso aberto.

Proposição 4.6.1 *Sejam P e Q processos. Tem-se $P \dot{\sim} Q$ se e só se $\mathcal{D}_{fe}(P) = \mathcal{D}_{fe}(Q)$.*

Demonstração Na subsecção 2.3.2 já mostramos que o functor F satisfaz todas as condições da proposição 2.3.9, logo se tomarmos a coálgebra Proc e a função semântica $\mathcal{D}_{fe} : \mathsf{Proc} \to \mathbb{D}$, que é o único morfismo de (Proc, γ) na coálgebra final $(\mathbb{D}, \chi)$, o resultado segue de imediato. □

A bissimilaridade não é uma congruência, porque não é preservada pelo prefixo de entrada, para obtermos uma congruência temos de exigir que os os processos sejam bissimilares para todas as substituições de nomes. A maior congruência contida na bissimilaridade é então definida do modo que se segue.

Definição 4.6.2 Dois processos P e Q são fortemente congruentes, e escreve-se $P \sim Q$, se $P\theta \dot{\sim} Q\theta$ para toda a substituição θ. □

Podemos agora enunciar a proposição que relaciona a congruência com a equivalência semântica aberta.

Proposição 4.6.3 *Sejam P e Q processos. Tem-se $P \sim Q$ se e só se $\mathcal{D}_{ab}(P) = \mathcal{D}_{ab}(Q)$.*

Demonstração Por definição $P \sim Q$ se $P\theta \dot{\sim} Q\theta$ para toda a substituição θ o que pela proposição anterior é o mesmo que $\mathcal{D}_{fe}(P\theta) = \mathcal{D}_{fe}(Q\theta)$, que por sua vez é equivalente a termos $\mathcal{D}_{ab}(P)\theta = \mathcal{D}_{ab}(Q)\theta$ donde, pela definição de função, $\mathcal{D}_{ab}(P) = \mathcal{D}_{ab}(Q)$. □

Capítulo 5

Considerações finais

Finalizamos este trabalho com algumas considerações sobre os objectivos atingidos e os possíveis desenvolvimentos que se poderão seguir.

As contribuições principais deste trabalho são: por um lado mostrar que abordagens, matematicamente mais simples que as tradicionais, podem ser usadas na definição da semântica de linguagens com concorrência e também de linguagens com mobilidade; por outro lado completar alguns aspectos da semântica da mobilidade para os quais as propostas actualmente existentes apresentam limitações.

Começamos por apresentar uma compilação de resultados necessários no decorrer do trabalho. A maior parte desses resultados fazem parte da teoria de base ou são adaptações, ao nosso contexto, de trabalhos de outros autores.

Seguidamente, usou-se a linguagem $\mathcal{L}_{syn}$ para mostrar como os cfe's podem ser usados, com vantagem sobre os espaços métricos, na semântica da concorrência e como a abordagem coalgébrica também se apresenta bastante adequada para lidar com os fenómenos da concorrência. Neste caso a complexidade de aplicação dos dois métodos não apresenta grandes desequilíbrios. Note-se que, em ambos os casos, a obtenção da semântica operacional resulta directamente do sistema de transições enquanto que com as técnicas métricas, [BV96], é necessário recorrer a um ponto fixo. A utilização dos cfe's para

a construção da semântica denotacional está assente sobre a definição dos operadores auxiliares (secção 3.2.5), cuja definição é intuitiva. Na definição coalgébrica da semântica denotacional evitamos a utilização de técnicas de ponto fixo pois os resultados são obtidos por coindução. Recorremos à técnica dos cfe's apenas para o cálculo do ambiente ρ_D, associado à declaração D, por esta solução se apresentar mais imediata. Estamos, no entanto, convencidos que seria possível atingir os objectivos utilizando uma abordagem coalgébrica pura, o que poderemos vir a fazer no futuro. Ainda na técnica coalgébrica, a obtenção dos domínios é mais intuitiva (usamos $\mathcal{P}_{fin}$ em vez de $\mathcal{P}_{co}$).

Mostramos, depois, a grande capacidade de expressão das técnicas coalgébricas, aliadas às dos conjuntos nominais, para estudar sistemas móveis baseados na manipulação de nomes. A abordagem coalgébrica que utilizamos tem alguma influencia de [Sta96], mas, no nosso caso, como temos uma coálgebra final os resultados são obtidos por coindução, o que simplifica o processo. Na definição da semântica denotacional, os operadores mais elementares não tiveram o mesmo tratamento coalgébrico que a composição paralela, a restrição e a replicação. O motivo para esta opção foi que, embora o tratamento coalgébrico dessas operações não levantasse qualquer problema técnico, recorrendo à utilização de χ^{-1} obtém-se resultados quase imediatos e evita-se sobrecarregar as demonstrações coindutivas com mais casos para verificar. Ainda no capítulo quarto, verificamos que a relação de equivalência entre processos, induzida pelas semânticas, coincide com o conceito geral de bissimilaridade forte, no caso fechado e com a congruência forte no caso aberto.

Optou-se por não apresentar a definição da semântica do cálculo-π em termos de cfe's, nos moldes em que foi feita para $\mathcal{L}_{syn}$, porque os estudos feitos nesse sentido conduziram-nos a expressões para as funções auxiliares demasiados complexas e às quais não foi possível, em tempo útil, darmos um tratamento simples e claro. Outro dos prosseguimentos deste trabalho poderá

passar por procurar chegar a uma formulação mais simples para definir a semântica de linguagens com mobilidade em termos de cfe's, o que poderá passar por explorar técnicas diferentes das utilizadas no caso de $\mathcal{L}_{syn}$.

Bibliografia

[AC98] Roberto M. Amadio e Pierre-Louis Curien. *Domains and Lambda-Calculi*, capítulo Functions and processes, páginas 421–435. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

[ACS96] Roberto M. Amadio, Ilaria Castellani e Davide Sangiorgi. On bisimulations for the asynchronous π-calculus. Em Ugo Montanary e Vladimiro Sassone, editores, *Proc. 7th International Conference CONCUR'96: Concurrency Theory*, volume 1119 de *Lecture Notes in Computer Science*, páginas 147–162. Springer, 1996.

[Acz88] Peter Aczel. *Non-well-founded sets*. Número 14 em Center for the Study of Languages and Information Lecture Notes. Stanford, 1988.

[Acz97] Peter Aczel. Lectures on semantics: The initial algebra and final coalgebra perspectives. Em H. Schwichtenberg, editor, *Logic of Computation*. Springer- Verlag, 1997. Lectures for the 1995 Marktoberdorf School on Logic of Computation.

[AM88] Peter Aczel e N. Mendler. A final coalgebra theorem. Em D. Pitts, D. Rydeheard, P. Dybjer e A. Poigne, editores, *Proc. Category Theory and Computer Science*, volume 389 de *Lecture Notes in Computer Science*, páginas 357–365. Springer, 1988.

[AR89] P. America e J. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39:343–375, 1989.

[BA90] M. Ben-Ari. *Principles of concurrent and distributed programming*. International Series in Computing Science. Prentice Hall International, 1990.

[Bal00] Michael Baldamus. Compositional constructor interpretation over coalgebraic models for the π-calculus. Em H. Reichel, editor, *Proc. Coalgebraic Methods in Computer Science, CMCS'2000*, volume 33 de *Electronic Notes in Theoretical Computer Science*, páginas 11–39. Elsevier, 2000.

[Ban22] S. Banach. Sur les opérations dans les ensembles abstraits et leurs applications aux equations intégrales. *Fundamenta Mathematicae*, (3):123–181, 1922.

[BB92] Gérard Berry e Gérard Boudol. The Chemical Abstract Machine. *Theoretical Computer Science*, 96:217–248, 1992.

[BCG97] Robert Bjornson, Nicholas Carriero e David Gelernter. From weaving threads to untangling the Web: a view of coordination from Linda's perspective. Em David Garlang e Daniel Le Métayer, editores, *Coordination Languages and Models: Proc. Second International Conference, Coordination'97*, páginas 1–17. Springer, 1997.

[Ber98] George M. Bergman. *An Invitation to general algebra and universal constructions*. Henry Helson, 1998.

[BJ93] Koen De Bosschere e Jean-Marie Jacquet. Multi-prolog: Definition, operational semantics and implementations. Em Warren

D.S., editor, *Proceedings of the 10th Int. Conference on Logic Programming*, páginas 299–313, Budapest, June 1993. MIT Press.

[BJB94] Koen De Bosschere, Jean-Marie Jacquet e Antonio Brogi, editores. *Proc. of ICLP'94 Post-Conference Workshop on Process-Based Parallel Logic Programming*, Santa Margherita Ligure - Italy, Junho 1994.

[BL95] Gérard Boudol e Cosimo Laneve. λ-calculus, multiplicities and the π-calculus. Rapport de recherche 2581, INRIA, Junho 1995.

[Bou92] Gérard Boudol. Asynchrony and the π-calculus. Research Report 1702, INRIA, 1992.

[BV96] Jaco De Bakker e Eric De Vink. *Control Flow Semantics*. Foundations of Computing Series. The MIT Press, 1996.

[BvW98] Ralph-Johan Back e Joakim von Wright. *Refinement calculus: a systematic introduction*. Graduate texts in computer science. Springer, 1998.

[Car99] Luca Cardelli. Mobility and security. Lecture notes for the Marktoberdof Summer School, 1999.

[CG89] Nicholas Carriero e David Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, Abril 1989.

[CG98] Luca Cardeli e Andrew Gordon. Mobile ambients. Em *Proc. Foundations of Software Science and Computation Structures (FoSSaCS'98)*, volume 1378 de *Lecture Notes in Computer Science*, páginas 140–155. Springer-Verlag, 1998.

[CM88] K. Mani Chandy e Jayadev Misra. *Parallel programming design: a foundation*. Addison-Wesley, 1988.

[CS00] Gian Luca Cattani e Peter Sewell. Models for name-passing processes: Interleaving and causal. Relatório Técnico 505, Computer Laboratory, University of Cambridge, Setembro 2000. 42pp.

[CSW97] Gian Luca Cattani, Ian Stark e Glynn Winskel. Presheaf models for the π-calculus. Em *Category Theory and Computer Science: Proceedings of the 7th International Conference CTCS '97*, número 1290 em Lecture Notes in Computer Science, páginas 106–126. Springer-Verlag, 1997.

[CSW00] Gian Luca Cattani, Ian Stark e Glynn Winskel. Models for name-passing processes: Interleaving and causal. Setembro 2000.

[DJ95] Herbert Dreshem e Michael Jipping. *Programming languages: structures and models.* The PWS series in Computer Science. International Thomson Publishing, 1995.

[FGL+96] Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget e Didier Rémy. A calculus of mobile agents. Em Ugo Montanary e Vladimiro Sassone, editores, *Proc. 7th International Conference CONCUR'96: Concurrency Theory*, volume 1119 de *Lecture Notes in Computer Science*, páginas 406–421. Springer, 1996.

[FMQ95] GianLuigi Ferrari, Ugo Montanary e Paola Quaglia. The weak late π-calculus semantics as observational equivalence. Em ?, editor, *Proc. 6th International Conference CONCUR'95*, volume 962 de *Lecture Notes in Computer Science*, páginas 57–71. Springer, 1995.

[FMS96] M. Fiori, E. Moggi e D. Sangiorgi. A fully-abstract model for the π-calculus. Em *LICS'96 Conference Proc.* IEEE, 1996.

[Gel85] David Gelernter. Generative communication in Linda. *ACM Transactions on programming languages and systems*, 7(1):80–112, Janeiro 1985.

[GP99] M. J. Gabbay e A. M. Pitts. A new approach to abstract syntax involving binders. Em *Proc. 14th Annual IEEE Symposium on Logic in Computer Science*, páginas 214–224. IEEE Computer Society Press, Washington, DC, 1999.

[GP02] M. J. Gabbay e A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002. Special issue in honour of Rod Burstall.

[Gru97] Jozef Gruska. *Foundations of Computing*. International Thomson Computer Press, 1997.

[GZ97] David Gelernter e Lenore Zuck. On what Linda is: Formal description of Linda as a reactive system. Em David Garlang e Daniel Le Métayer, editores, *Coordination Languages and Models: Proc. Second International Conference, Coordination'97*, páginas 187–204. Springer, 1997.

[Han94] Chris Hankin. *Lambda Calculi A Guide for Computer scientists*. Graduate Texts in Computer Science. Clarendon Press - Oxford, 1994.

[Han98] Chris Hankin, editor. *Programming Languages and Systems, Proc. 7th European Symposium on Programming, ESOP'98, Lisbon-Portugal*, volume 1381 de *Lecture Notes in Computer Science*. Springer, 1998.

[HC94] Jonathan M. D. Hill e Keith Clarke. An introduction to category theory, category theory monads, and their relationship to

functional programming. Relatório Técnico QMW-DCS-681, Department of Computer Science, Queen Mary & Westfield College, Agosto 1994.

[Hen02] Matthew Hennessy. A fully abstract denotational semantics for the pi-calculus. *Theoretical Computer Science*, 279(1-2):53–89, 2002.

[Hoa85] C. A. R. Hoare. *Communicating Sequencial Processes*. Series in Computer Science. Prentice-Hall International, 1985.

[Hon00] Kohei Honda. Elementary structures in process theory (1): Sets with renaming. *Journal of Mathematical Structures in computer Science*, 10:617–663, Outubro 2000.

[HS98] John Harris e Horst Stocker. *Handbook of mathematics and computational science*. Springer, 1998.

[HY94a] Kohei Honda e Nobuko Yoshida. Combinatory representation of mobile processes. Em *Proc. of the 21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, páginas 348–360, 1994.

[HY94b] Kohei Honda e Nobuko Yoshida. Replication in concurrent combinators. Em *Proc. of TACS'94*, volume 789 de *Lecture Notes in Computer Science*, páginas 786–805. Springer-Verlag, 1994.

[JM94] Jean-Marie Jacquet e Luís Monteiro. Towards resource handling in logic programming: the PPL framework and its semantics. Em Koen de Bosschere, Jean-Marie Jacquet e Antonio Brogi, editores, *Proc. of ICLP'94 Post-Conference Workshop on Process-Based Parallel Logic Programming*, páginas 39–54, 1994.

[JR97] Bart Jacobs e Jan Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, (62):222–259, 1997.

[Len98] Marina Lenisa. *Themes in final semantics*. Tese de Doutoramento, Università di Pisa - Udine, Março 1998.

[Lop99] Luís Lopes. *On the Design and Implementation of a virtual Machine for Process Calculi*. Tese de Doutoramento, Departamento de Ciência de Computadores - Faculdade de Ciências da Universidade do Porto, Outubro 1999.

[Lou93] Kenneth Louden. *Programming Languages: principles and practice*. PWS-KENT series in computer science. PWS Publishing Company, 1993.

[Mes96] José Meseguer. Rewriting logic as a semantic framework for concurrency: a progress report. Em Ugo Montanary e Vladimiro Sassone, editores, *Proc. 7th International Conference CONCUR'96: Concurrency Theory*, volume 1119 de *Lecture Notes in Computer Science*, páginas 331–372. Springer, 1996.

[Mil80] Robin Milner. *A calculus of Communicating Systems*. Número 92 em Lecture Notes in Computer Science. Springer, 1980.

[Mil89] Robin Milner. *Comunication and concurrency*. International Series in Computing Science. Prentice Hall International, 1989.

[Mil90] Robin Milner. Operational and algebraic semantics of concurrent processes. Em J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, páginas 1201–1243. Elsevier, 1990.

[Mil93] Robin Milner. The polyadic π-calculus: a tutorial. Em F.L. Bauer, W. Brauer e H. Schwichtenberg, editores, *Logic and Algebra of Specification*, NATO ASI Series, páginas 203–246. Springer-Verlag, Berlin, 1993.

[Mil99] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[Mon98] Luís Monteiro. Semantic domains based on sets with families of equivalences. Em B. Jacobs, L. Moss, H. Reichel e J. Rutten, editores, *Coalgebraic Methods in Computer Science(CMCS'98)*, número 11 em Electronic Notes in Theoretical Computer Science, páginas 73–106. Elsevier, 1998.

[Mon99] Luís Monteiro. All nontrivial set functors give rise to final coalgebras. Relatório técnico, FCT - Universidade Nova de Lisboa, 1999.

[Mon00] Luís Monteiro. Observation systems. Em H. Reichel, editor, *Proc. Coalgebraic Methods in Computer Science CMCS'2000*, volume 33 de *ENTCS*, páginas 265–279. Elsevier Science, 2000.

[Mon03] Luís Monteiro. Notes on nominal sets. Unpublished notes. DI-FCT Universidade Nova de Lisboa, 2003.

[MP92] Zohar Manna e Amir Pnueli. *The temporal logic of reactive and concurrent systems: specification*. Springer-Verlag, 1992.

[MP95] Zohar Manna e Amir Pnueli. *Temporal verification of reactive systems: safety*. Springer-Verlag, 1995.

[MP98] Luís Monteiro e António Porto. Entailment-based actions for coordination. *Theoretical Computer Science*, (192):256–286, 1998.

[MPW92] Robin Milner, Joachim Parrow e David Walker. A calculus of mobile processes, part I/II,. *Information and Computation*, (100):1–77, 1992.

[MPW93] Robin Milner, Joachim Parrow e David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.

[MR97] Peter J. McCann e Gruia-Catalin Roman. Mobile UNITY coordination constructs applied to packet forwarding for mobile hosts. Em David Garlang e Daniel Le Métayer, editores, *Coordination Languages and Models: Proc. Second International Conference, Coordination'97*, páginas 338–354. Springer, 1997.

[NFP97] Rocco De Nicola, GianLuigi Ferrari e Rosario Pugliese. Coordinating mobile agents via blackboards and access rights. Em David Garlang e Daniel Le Métayer, editores, *Coordination Languages and Models: Second International Conference, Coordination'97 Proc.*, páginas 220–237. Springer, 1997.

[NN92] Hanne Riis Nielson e Flemming Nielson. *Semantics with applications - A formal introduction.* John Wiley & Sons, 1992.

[Pal97] Catuscia Palamidessi. Comparing the expressive power of the syncronous and the asyncronous π-calculus. Em *Proc. 24th ACM Symposium on Principles of Programming Languages (POPL)*, páginas 256–265. ACM, 1997.

[Par96] Alan Parkes. *An introduction to computable languages & abstract machines.* International Thomson Computer Press, 1996.

[Par01] Joachim Parrow. An introduction to the π-calculus. Em Bergstra, Ponse e Smolka, editores, *Handbook of Process Algebra.* Elsevier, 2001.

[Pit01] Andrew M. Pitts. Nominal logic: A first order theory of names and binding. Em N. Kobayashi e B. C. Pierce, editores, *Fourth International Symposium on Theoretical Aspects of Computer Software(TACS 20001), Sendai, Japan*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2001.

[PT96] Benjamin C. Pierce e David N. Turner. Pict: A programming language based on the pi-calculus. Maio 1996. Draft report.

[PV96] António Porto e Vasco T. Vasconcelos. Truth and action osmosis: the TAO computation model. Em J.-M. Andreoli, C. Hankin e D. Le Métayer, editores, *Coordination Programming: Mechanisms, Models, and Semantics*, páginas 65–97. ICP, 1996.

[QW00] Paola Quaglia e David Walker. On synchronous and asyncronous mobile processes. Em *Proc. 3rd. International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2000*, páginas 283–296. Springer, 2000.

[Rav00] António Ravara. *Typing non-uniform concurrent objects*. Tese de Doutoramento, Instituto Superior Técnico - Universidade Técnica de Lisboa, Agosto 2000.

[RT94] J. Rutten e D. Turi. Initial algebra and final coalgebra semantics for concurrency. Em W.P. de Roever J.W. de Bakker e G. Rozenberg, editores, *Proc. REX School: A decade of concurrency*, número 803 em Lecture Notes in Computer Science, páginas 530–582. Springer, Berlin, 1994.

[Rut95] J. Ruten. A calculus of transition systems (towards universal co-algebra). Em A. Ponse, M. de Rijke e Y. Venema, editores, *Modal Logic and Process Algebra, A Bissimulation Perspective*, número 53 em CSLI Lecture Notes, páginas 231–256. CSLI Publications, Stanford, 1995.

[Rut96] J. Rutten. Universal coalgebra: a theory of systems. Research Report CS-R9652, CWI, Amsterdam, 1996.

[Rut98] Jan Rutten. Automata and coinduction (an exercise in coalgebra). Em David Sangiorgi e Robert de Simone, editores, *Proc.*

CONCUR'98 Concurrency Theory: 9th International Conference, volume 1466 de *Lecture Notes in Computer Science*, páginas 194–218. Springer, 1998.

[Rut99] J. Rutten. Automata, power series, and coinduction: Taking input derivates seriously(extended abstract. SEN R9901, CWI, Janeiro 1999.

[Rut00] J. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. (Revised and extended version of Technical Report CS-R9652, 1996).

[San92] Davide Sangiorgi. *Expressing Mobility in Process Algebras:First-order and Higher-order paradigms*. Tese de Doutoramento, Departement of Computer Science, University of Edinburgh, 1992.

[San95] Davide Sangiorgi. Bisimulation for higher-order process calculi. Rapport de recherche, INRIA, Abril 1995.

[San99] Davide Sangiorgi. Asynchronous process calculi: the first-order and higher-order paradigms (tutorial). 1999. 39 pp.

[Sar93] Vijay Saraswat. *Concurrent constraint programming*. ACM Doctoral Dissertation Awards. The MIT Press, 1993.

[Sch97] Fred Schneider. *On concurrent programming*. Graduate Texts in Computer Science. Springer, 1997.

[Sem96] Laura Semini. *Refiniment in tuple space languages*. Tese de Doutoramento, Università Degli Studi di Pisa, 1996.

[Sew95] Peter Michael Sewell. *The Algebra of Finite State Processes*. Tese de Doutoramento, University of Edinburgh, Outubro 1995. Department of Computer Science technical report CST-118-95, also published as LFCS-95-328.

[Sew99] Peter Sewell. A brief introduction to applied π, Janeiro 1999. Lecture notes for the Mathfit Instructional Meeting on Recent Advances in Semantics and Types for Concurrency: Theory and Practice, July 1998.

[Sew00] Peter Sewell. Applied π – a brief tutorial. Relatório Técnico 498, Computer Laboratory, University of Cambridge, Agosto 2000. 65pp.

[Sha89] Ehud Shapiro. The family of concurrent logic programming languages. *ACM Computing Surveys*, 21(3):413–510, Setembro 1989.

[SM97] Laura Semini e Luís Monteiro. A verification calculus for the TAO coordination model. Em *Proc. of the Internetional Workshop on Verification, Model Checking and Abstract Interpretation*. Port Jefferson, Long Island N.Y., Outubro 1997.

[SP82] M. Smyth e G. Plotkin. The categorial theoretic solution of recursive domain equations. *Journal of Computing*, 4(11):761–783, 1982.

[SRP90] Vijay Saraswat, Martin Rinard e Prakash Panangaden. Semantic foundations of concurrent constraint programming. *ACM*, páginas 333–345, 1990.

[Sta94] Ian Stark. *Names and Higher-Order Functions*. Tese de Doutoramento, University of Cambridge, Dec. 1994. Also published as Technical Report 363, University of Cambridge Computer Laboratory.

[Sta96] Ian Stark. A fully abstract domain model for the π-calculus. Em *LICS'96 Conference Proceedings*, páginas 36–42. IEEE Press, 1996.

[SW01] Davide Sangiorgi e David Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[Vas94] Vasco T. Vasconcelos. *A process-calculus approach to typed concurrent objects*. Tese de Doutoramento, KEIO, 1994.

[VP96] Björn Victor e Joachim Parrow. Constraints as processes. Em Ugo Montanary e Vladimiro Sassone, editores, *Proc. 7th International Conference CONCUR'96: Concurrency Theory*, volume 1119 de *Lecture Notes in Computer Science*, páginas 389–405. Springer, 1996.

[VT93] Vasco T. Vasconcelos e Mario Tokoro. A typing system for a calculus of objects. Em *1st ISOTAS*, volume 472 de *Lecture Notes in Computer Science*, páginas 460–474. SPRINGER, Novembro 1993.

[Wal91] R. F. C. Walters. *Categories and Computer Science*, volume 28 de *Cambridge Computer Science Texts*. Cambridge University Press, 1991.

[Wal95] David Walker. Objects in the π-calculus. *Information and Computation*, (116):253–271, 1995.

[Win93] Glynn Winskel. *The formal semantics of programming languages: An introduction*. Foundations of Computing. The MIT Press, 1993.

[Wol99] Uwe Wolter. A coalgebraic introduction to CSP. 1999. Elsevier Science.