UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

LÓGICA TEMPORAL DE OBJECTOS COM TEMPO EXPLÍCITO

João Fernando da Conceição Gomes
(licenciado)

Dissertação para Obtenção do Grau de Mestre em
Matemática Aplicada

Orientadora:
Doutora Cristina Sernadas

Júri:
Doutor Amílcar Sernadas
Doutor José Valença
Doutora Cristina Sernadas

Dezembro de 1995

João Fernando da Conceição Gomes

Departamento de Matemática — Secção de Ciência da Computação

Mestrado em Matemática Aplicada

Orientadora: Professora Doutora Cristina Sernadas

Dezembro de 1995

Lógica Temporal de Objectos com Tempo Explícito

Resumo

Define-se uma lógica temporal de base proposicional que permite descrever propriedades de objectos que envolvam tempo explícito. Considera-se que cada objecto tem o seu relógio interno próprio, caracterizado pelo seu período. Assim, o tempo é relativo e intervalar, tendo cada intervalo um comprimento igual ao período do relógio. A lógica adoptada contém variantes dos operadores temporais usuais de modo a referirem explicitamente o tempo. Apresenta-se um sistema dedutivo baseado em tableaux, que se prova ser correcto, completo e decidível para raciocínio local com um objecto. Introduzem-se ainda técnicas de agregação de objectos e faz-se a extensão do sistema dedutivo para raciocinar sobre agregações.

Object Temporal Logic with Explicit Time

Abstract

A propositional temporal logic for reasoning with objects is presented. In order to work with explicit time, time values are introduced into the logic. It is considered that each object includes a local clock to measure time. Time is interpreted as being relative and interval-based, with intervals of equal length. In order to work with time, temporal operators are associated with time values. A sound, complete and decidable tableaux system is presented. Future and past cycles are introduced to obtain a decidable algorithm. Aggregation techniques for objects are introduced, as well as semantics and a tableaux system.

Nome: João Fernando da Conceição Gomes

Título: Lógica Temporal de Objectos com Tempo Explícito

Palavras Chave

Especificação centrada em objectos
Agregação
Lógica temporal com tempo explícito
Sistemas de tableaux
Correcção
Completude
Decidibilidade

Keywords

Object-oriented specification
Aggregation
Temporal logic with explicit time
Tableaux systems
Soundness
Completeness
Decidability

Agradecimentos

À Professora Cristina Sernadas, pelo grande interesse demonstrado na orientação deste trabalho e por estar sempre disponível para indispensáveis discussões. A realização deste trabalho não teria sido possível sem o seu apoio.

Ao Professor Amílcar Sernadas, pelas valiosas sugestões e pela oportunidade de pertencer ao seu grupo de investigação.

À Maria João e aos colegas da Secção de Ciência da Computação, pelo apoio sempre demonstrado.

À Ana, por tudo.

Índice

1 Introdução **1**

2 Introdução do Tempo **5**

2.1 Introdução 5

2.2 Domínios de Tempo 5

3 Lógica de Objectos: Sintaxe e Semântica **12**

3.1 Introdução 12

3.2 Sintaxe de Objectos 13

3.2.1 Assinatura e Termos 13

3.2.2 Fórmulas 20

3.3 Semântica 24

3.3.1 Estrutura de Interpretação 24

3.3.2 Relação de Satisfação 28

4 Sistema Dedutivo: Decidibilidade **35**

4.1 Introdução 35

4.2 Fórmulas Prefixadas 36

4.2.1 Sintaxe 36

4.2.2 Semântica 38

4.3 Sistema de Tableaux 41

4.4 Decidibilidade 57

4.4.1 Correcção e Completude 57

4.4.2 Algoritmo de Decisão 60

5 Agregação de Objectos **73**

5.1 Introdução 73

5.2 Categoria de Especificação de Objectos 74

5.3 Comunidade de Objectos 79

5.3.1 Sintaxe e Semântica 80

5.3.2 Sistema Dedutivo 85

6 Conclusões **86**

Bibliografia **88**

Apêndice **94**

A Categorias **94**

A.1 Categorias 94

A.1.1 Conceitos 94

A.1.2 Exemplos 98

A.2 Functores 99

Capítulo 1

Introdução

O uso da lógica temporal como um formalismo para especificar o comportamento de sistemas ao longo do tempo foi proposto em [Pnueli 77, Sernadas 80]. A lógica temporal permite traduzir de um modo natural os requisitos temporais qualitativos dos sistemas, recorrendo aos seus operadores temporais para o futuro (X, U) e para o passado (Y, S). Existem várias variantes da lógica temporal [Emerson 90], apresentando diferenças a nível sintáctico e semântico. Dois casos particulares são a lógica temporal de primeira ordem e a lógica temporal proposicional, [Bell e Machover 77].

A semântica usual para a lógica temporal é constituída por estruturas de Kripke. Estas estruturas são compostas por um conjunto de estados, uma relação binária sobre os estados que possibilita navegar de estado para estado, e uma função de valoração que indica quais as propriedades verificadas em cada estado.

É possível raciocinar sintacticamente sobre as fórmulas temporais através de sistemas dedutivos. Existem duas grandes classes de sistemas dedutivos: os sistemas axiomáticos e os operacionais. Um sistema axiomático é composto por um conjunto de fórmulas intituladas axiomas e por um conjunto de regras de derivação. Um sistema operacional é consistuído por conjuntos de conjuntos de fórmulas intitulados absurdos e por regras de inferência.

Um dos sistemas dedutivos é baseado em tableaux [Beth 56] e [Hintikka 55]. Posteriormente, [Smullyan 68] definiu os conceitos fundamentais para estes sistemas. Existem dois tipos de tableaux, declarativo [Fischer e Ladner 79, Sistla e Clarke 85] e incremental [Pnueli e Sherman 81, Ben-Hari *et al* 83, Manna e Wolper 84]. Os tableaux declarativos são mais fáceis de interpretar e de analisar, sendo preferíveis quando se pretende efectuar uma implementação.

O sistema de tableaux para a lógica temporal proposicional e para a lógica temporal de primeira ordem é correcto e completo. No entanto, o primeiro é decidível, sendo o segundo semi-decidível. Em [Ben-Hari *et al* 83, Manna e Wolper 84, Kesten *et al* 93] são apresentados algoritmos de decisão para a lógica temporal proposicional baseados em tableaux incrementais.

Os sistemas orientados a objectos têm vindo a despertar grande interesse na comunidade científica. Desde que a linguagem `SIMULA` [Dahl *et al* 67] foi desenvolvida, a definição de objecto tem vindo a evoluir, sendo nesta dissertação adoptada a definição apresentada em [Sernadas *et al* 89, Sernadas e Ehrich 91]. Actualmente encontram-se definidas as linguagens de objectos `OBLOG` [Costa *et al* 89, Sernadas *et al* 91, Sernadas *et al* 92b, Sernadas *et al* 92a, Sernadas *et al* 95] com uma versão textual `GNOME` [Sernadas e Ramos 94, Ramos 94, Ramos e Sernadas 95a, Ramos e Sernadas 95b] e `TROLL` [Jungclaus *et al* 95]. A filosofia de objecto tem sido utilizada em sistemas de informação e em linguagens de programação, entre outras aplicações. Um objecto contém atributos, executa acções e apresenta um comportamento. É usual recorrer à lógica temporal para raciocinar com objectos. Na lógica temporal de primeira ordem, os tipos de dados abstractos permitem parametrizar as acções, sendo as fórmulas temporais utilizadas para descrever propriedades satisfeitas pelo objecto [Sernadas *et al* 95, Sernadas e Sernadas 94]. Com a lógica temporal proposicional, as acções não apresentam parâmetros e os atributos são vistos como expressões booleanas [Gouveia *et al* 93, Gouveia 92]. A lógica OSL é apresentada por via axiomática em [Sernadas *et al* 95, Sernadas e Sernadas 94] incluindo mecanismos de raciocínio para objectos e agregação de objectos. Em [Li e Sernadas 91, Gouveia 92, Gouveia *et al* 93] são apresentados sistemas baseados em tableaux para raciocinar localmente sobre os objectos.

A necessidade de trabalhar com o tempo de forma explícita tem vindo a aumentar como consequência de avanços tecnológicos, especialmente nas áreas de sistemas de controlo e de segurança. Como reflexo desta necessidade, aspectos de tempo explícito têm vindo a ser introduzidos em vários formalismos, como em redes de Petri [Ostroff 89, Felder *et al* 94], em autómatos [Dill 89, Alur e Dill 90], em álgebras de processos [Lee *et al* 94] e em lógicas modais [Goldblatt 87] e temporais [Koymans 89, Koymans 90, Koymans 92].

Apesar de o tempo ser universal, vários formalismos têm sido desenvolvidos com o objectivo de trabalhar com o tempo. Independentemente da lógica onde o tempo irá ser inserido, existe a possibilidade de raciocinar sobre intervalos de tempo [Alur e Henzinge 91], quando é suficiente afirmar que uma fórmula é verificada entre dois valores de tempo; pode-se também raciocinar sobre instantes de tempo, quando é necessário indicar explicitamente o valor de tempo onde a fórmula é verificada. Os valores de tempo poderão ser inteiros ou reais. Por vezes recorre-se a um tempo absoluto, em que é usada uma recta para especificar todo o tempo; outras vezes considera-se um tempo relativo, onde cada momento do presente representa o início de uma recta de tempo. Em [Alur e Henzinge 90] podemos encontrar uma lógica temporal com tempo pontual com valores inteiros; [Böhm 93, Böhm e Sernadas 93, Koymans 90, Alur e Dill 90, Henzinger 91] apresentam o tempo como sendo pontual, mas real. Em [Alur e Henzinge 91] o tempo é intervalar com valores reais.

No contexto das lógicas temporais, é possível ainda considerar tempo linear (existe um único futuro) ou ramificado (onde poderão existir vários futuros). O tempo pode ser inserido nas fórmulas de modo a limitar o âmbito dos operadores temporais: ao associar intervalos (valores) de tempo aos operadores temporais, obriga-se a fórmula abrangida pelo operador a ser verificada nesse intervalo (valor) [Pnueli e Harel 88, Koymans 90, Koymans 92]. O tempo também poderá ser incorporado na lógica recorrendo a uma variável de tempo que indique explicitamente o tempo em cada estado [Pnueli e Harel 88, Ostroff 89].

Esta dissertação tem como objectivo principal definir uma lógica temporal com tempo explícito para o raciocínio de objectos, tomando como base a lógica apresentada em [Sernadas e Sernadas 94, Sernadas *et al* 95]. Ao fazer a extenção da lógica apresentada em [Sernadas e Sernadas 94], por inserção do tempo, torna-se possível descrever propriedades sobre os objectos explicitamente com valores de tempo e considerar acções com duração de execução. O tempo será relativo, com valores racionais e serão considerados intervalos de tempo com um comprimento fixo. Os operadores temporais serão enriquecidos com valores de tempo, seguindo [Koymans 89, Koymans 92]. A lógica temporal será de base proposicional, permitindo definir um algoritmo de decisão.

Em [Böhm 93, Böhm e Sernadas 93] é apresentada uma extensão da lógica de especificação de objectos [Sernadas *et al* 95] com tempo pontual, absoluto e com valores reais. A associação de valores de tempo é feita unicamente nas fórmulas temporais que descrevem ocorrências de acções.

No segundo capítulo é introduzida a noção de tempo e definido o domínio de tempo. Definem-se ainda relações, operações e aplicações essenciais para manipular o tempo.

No capítulo seguinte definem-se a sintaxe e a semântica da lógica temporal com tempo explícito para objectos. Apresenta-se a noção de assinatura de objecto, e em seguida, as definições dos termos e das fórmulas temporais. A semântica escolhida será baseada em estruturas de Kripke, sendo apresentadas uma relação de satisfação e uma consequência semântica.

No quarto capítulo é definido um sistema dedutivo e faz-se a verificação da sua correcção e da sua completude. Introduz-se à partida a noção de fórmulas prefixadas e definem-se a sintaxe e a semântica dessas fórmulas. O sistema dedutivo é baseado num sistema de tableaux, sendo definidos os absurdos e as regras de inferência do sistema. Finalmente provam-se a correcção, a completude e a decidibilidade do sistema dedutivo, apresentando-se ainda um algoritmo de decisão.

No quinto capítulo é efectuada a agregação de objectos. A categoria das assinaturas e a categoria das especificações de objectos são definidas. No final do capítulo é apresentada a definição de comunidade de objecto, com a sua sintaxe e semântica, bem como um sistema dedutivo para uma comunidade baseado no quarto capítulo.

No final da dissertação, em apêndice, encontram-se algumas noções sobre categorias que foram utilizadas ao longo dos capítulos.

Capítulo 2

Introdução do Tempo

2.1 Introdução

Inicialmente o tempo com que se pretende trabalhar é relativo, contínuo e pontual. No entanto, vamos permitir dois enfranquecimentos nos nossos valores de tempo. A recta do tempo irá ser repartida em intervalos de igual comprimento. Esta repartição irá permitir usufruir das vantagens do tempo discreto sem se deixar de trabalhar com valores reais. O outro enfraquecimento consistirá em considerar somente os racionais como valores de tempo. Em [Ladkin 87a, Ladkin 87b, Li e Carmo 93] é demonstrado que o conjunto formado por tuplos $\langle ano, mês, dia, hora, min, \ldots \rangle$ onde ano toma valores inteiros, $mês$ toma valores naturais desde 1 até 12, etc, é isomorfo aos racionais. Para a lógica de objectos desenvolvida neste trabalho o tuplo apresentado é suficiente para trabalhar com o tempo. Assim, optamos por restringir os valores de tempo aos racionais.

2.2 Domínios de Tempo

Definição 2.2.1 (Domínios de Valores de Tempo)
Seja q um racional não negativo ou ∞. Diz-se que o conjunto de racionais $\Delta = [0, q[\cap \mathbb{Q}$ é um domínio de valores de tempo.

□

Como vamos trabalhar com tempo relativo, é possível limitar superiormente o conjunto de valores de tempo sem daí tirar a conclusão de que o futuro e o passado estão limitados. Além disso, limitar superiormente um conjunto de valores de tempo facilita a especificação de certos objectos. Por exemplo, uma base de dados que só contenha informações com menos de dois anos é especificada mais facilmente se considerarmos um conjunto de valores de tempo equivalente a dois anos. Com esta opção obtemos três tipos de conjuntos de valores de tempo: os racionais não negativos, os racionais não negativos inferiores a um dado racional e o conjunto vazio. Este último caso ocorre quando se está interessado em trabalhar com a lógica temporal clássica.

Cada objecto contém um relógio interno. O relógio de um objecto é caracterizado pelo seu período que pode variar de objecto para objecto. Por exemplo, certos objectos poderão ter um período de 1 segundo, outros de 10 segundos, outros até de 1 dia ou mais. Este valor indica a velocidade de resposta do objecto a estímulos, uma vez que o objecto não consegue distinguir conjuntos de valores de tempo inferiores ao comprimento do seu período. Um processador de computador a 66 Mhz tem um relógio cujo período é o dobro do de um relógio dum processador a 33 Mhz. Uma base de dados central que só permite actualização no fim de cada dia poderá ter como período do seu relógio um racional que corresponde a 24 horas. Estes são exemplos de objectos em que são conhecidos os períodos dos relógios. Mas se pretendermos especificar um objecto cujo o período é desconhecido, não é possível escolher um racional para o caracterizar, pois tal seria fixar um valor para o período do seu relógio. Para tratar estes casos, introduzimos um novo símbolo. Esse novo símbolo será designado por ι e corresponde a completar os racionais positivos. A noção de conjunto de valores de tempo é agora estendida a dois novos conjuntos.

Notação 2.2.2 Sendo Δ um domínio de valores de tempo,

- $\Delta_\iota = \Delta \cup \{\iota\}$,
- $\Delta_\iota^+ = \Delta_\iota - \{0\}$.

□

No caso da lógica temporal clássica, Δ é o conjunto vazio e, portanto, Δ_ι é o conjunto singular $\{\iota\}$.

Uma vez introduzido o novo elemento ι, vamos indicar a sua relação com os restantes racionais do conjunto Δ. Optamos por considerar que ι é maior do que os restantes valores do conjunto. Assim temos as seguintes relações sobre Δ_ι:

Definição 2.2.3 (Relações Binárias)
Seja $\Delta = [0, q[\cap \mathbb{Q}$. Definimos as seguintes relações:

- $<_\iota \subseteq \Delta_\iota \otimes \Delta_\iota$

  diz-se que $\delta_1 <_\iota \delta_2$ sse

  - $\delta_1 \neq \iota$

    e

– se $\delta_2 \neq \iota$ então $\delta_1 < \delta_2$.

- $\leq_\iota \subseteq \Delta_\iota \otimes \Delta_\iota$

  diz-se que $\delta_1 \leq_\iota \delta_2$ sse se $\delta_1 \neq \delta_2$ então $\delta_1 <_\iota \delta_2$.

$\square$

É também possível definir duas operações básicas sobre o conjunto Δ_ι.

Definição 2.2.4 (Operações)
Seja $\Delta = [0, q[\cap \mathbb{Q}$. Definem-se as seguintes operações sobre Δ_ι:

- $+_\iota : \Delta_\iota \otimes \Delta_\iota \to \Delta_\iota$

  sejam $\delta_1, \delta_2 \in \Delta_\iota$,

$$\delta_1 +_\iota \delta_2 = \begin{cases} \delta_1 + \delta_2 & \text{se } \delta_1 \neq \iota, \delta_2 \neq \iota \text{ e } \delta_1 + \delta_2 < q \\ \iota & \text{caso contrário} \end{cases}$$

- $-_\iota : \Delta_\iota \otimes \Delta_\iota \to \Delta_\iota$

  sejam $\delta_1, \delta_2 \in \Delta_\iota$,

$$\delta_1 -_\iota \delta_2 = \begin{cases} \delta_1 - \delta_2 & \text{se } \delta_1 \neq \iota, \delta_2 \neq \iota \text{ e } 0 \leq \delta_1 - \delta_2 \\ 0 & \text{se } \delta_1 \neq \iota \text{ e se } \delta_2 \neq \iota \text{ então } \delta_1 - \delta_2 < 0 \\ \iota & \text{se } \delta_1 = \iota \end{cases}$$

$\square$

É necessário definir duas aplicações: a primeira efectua a multiplicação de um natural por um elemento do domínio de valores de tempo, a segunda efectua a divisão entre dois elementos do domínio de valores de tempo com quociente natural.

Definição 2.2.5 (Aplicações Auxiliares)
Seja $\Delta = [0, q[\cap \mathbb{Q}$. Definem-se recursivamente as seguintes aplicações:

- $\times_\iota : \Delta_\iota \otimes \mathbb{N}_o \to \Delta_\iota \cup \{0\}$

  sejam $\delta \in \Delta_\iota$ e $n \in \mathbb{N}_o$,

$$\delta \times_\iota n = \begin{cases} 0 & \text{se } n = 0 \\ \delta +_\iota (\delta \times_\iota (n-1)) & \text{se } n > 0 \end{cases}$$

- $\div_\iota : \Delta_\iota \otimes \Delta_\iota \nrightarrow I\!N_o$

  sejam $\delta_1, \delta_2 \in \Delta_\iota$,

$$\delta_1 \div_\iota \delta_2 = \begin{cases} 0 & \text{se } \delta_1 <_\iota \delta_2 \\ 1 + ((\delta_1 -_\iota \delta_2) \div_\iota \delta_2) & \text{se } \delta_2 \leq_\iota \delta_1 \end{cases}$$

□

Na multiplicação é necessário fazer a união de Δ_ι com zero. Sem esta união, a multiplicação estaria mal definida, uma vez que, no caso de Δ ser vazio, $\iota \times 0$ não pertenceria ao contradomínio. Ao longo desta dissertação, diz-se que o valor δ_2 é múltiplo de δ_1 sse existe um natural n tal que $\delta_1 \times_\iota n = \delta_2$.

A divisão é definida recursivamente recorrendo à adição e subtracção. Esta operação é parcial, não se encontrando definida quando o δ_1 é ι ou quando o δ_2 é zero. Esta parcialidade é um resultado imediato das propriedades subtracção.

Em seguida definimos o modo de calcular o maior múltiplo de um valor de tempo δ_1 menor que δ_2.

Definição 2.2.6 Sejam $\delta_1, \delta_2 \in \Delta_\iota$. O maior múltiplo de δ_1 menor que δ_2 é calculado do seguinte modo:

$$mul(\delta_1, \delta_2) = \begin{cases} 0 & \text{se } \delta_1 = 0 \\ \iota & \text{se } \delta_2 = \iota \\ \delta_1 \times_\iota (\delta_2 \div_\iota \delta_1) & \text{se } \delta_1 \neq 0 \text{ e } \delta_2 \neq \iota \end{cases}$$

□

A operação *mul* é definida com base na multiplicação e na divisão. A divisão determina a parte natural da divisão dos dois valores de tempo. Este natural é depois multiplicado pelo valor δ_1 para obter o maior múltiplo menor que δ_2. Se o valor δ_2 já é um múltiplo de δ_1, então o resultado da operação acima é o próprio δ_2. Uma vez que a divisão não se encontra definida para o caso em que δ_1 é zero e δ_2 é ι, foi necessário considerar estes dois casos em separado e atribuir-lhes o resultado que parece ser o mais natural.

De seguida estabelecemos uma relação de divisão entre dois valores de tempo.

Definição 2.2.7 (Relação de Divisão)
Sejam $\delta_1, \delta_2 \in \Delta_\iota$. A relação de divisão $|_\iota \subseteq \Delta_\iota \otimes \Delta_\iota$ é definida do seguinte modo:

$\delta_1 |_\iota \delta_2$ sse $mul(\delta_1, \delta_2) = \delta_2$.

□

Para definir a relação de divisão recorreu-se à operação mul para que a relação de divisão se encontre definida no caso em que δ_2 é ι.

Para finalizar este capítulo, iremos definir uma aplicação que determine o maior elemento do domínio de tempo cuja divisão por dois elementos do domínio de tempo previamente fixado seja inteira. Esta aplicação apresenta propriedades semelhantes às do máximo divisor comum para inteiros e será essencial para definir a agregação de especificações de objectos no capítulo 5.

Prop/Definição 2.2.8 (Máximo Divisor Comum Generalizado)
O máximo divisor comum generalizado, $mdc_g : \Delta_\iota \otimes \Delta_\iota \to \Delta_\iota$ é definido do seguinte modo:

$$mdc_g(\delta_1, \delta_2) = \begin{cases} \delta_1 & \text{se } \delta_2 = \iota \\ \delta_2 & \text{se } \delta_1 = \iota \\ \frac{mdc(n_1 d_2, n_2 d_1)}{d_1 d_2} & \text{se } \delta_1 = \frac{n_1}{d_1} \text{ e } \delta_2 = \frac{n_2}{d_2} \text{ para } n_1, n_2 \in I\!N_o \text{ e } d_1, d_2 \in I\!N \end{cases}$$

onde mdc é a aplicação que determina o máximo divisor comum entre dois naturais.

Demonstração:
Verifiquemos que a aplicação mdc_g está bem definida para quaisquer valores $\delta_1, \delta_2 \in \Delta_\iota$. Se $\delta_1 = \iota$ ou $\delta_2 = \iota$, o valor calculado por mdc_g é único.
Consideremos o caso em que $\delta_1 \neq \iota$ e $\delta_2 \neq \iota$, isto é, δ_1, δ_2 são números racionais não negativos. Existem dois naturais n_1, n_2 e dois naturais positivos d_1, d_2 tais que

$$\delta_1 = \frac{n_1}{d_1} \text{ e } \delta_2 = \frac{n_2}{d_2}.$$

Para estes valores o máximo divisor comum de $n_1 d_2$ e $n_2 d_1$ é único. Ao considerarmos naturais diferentes mas cujo quociente seja igual, obtem-se o mesmo resultado. Seja $m \in I\!N$:

$$mdc_g(\delta_1, \delta_2) = mdc_g\left(\frac{mn_1}{md_1}, \frac{n_2}{d_2}\right) =_{def\ mdc_g}$$

$$=_{def\ mdc_g} \frac{mdc(mn_1 d_2, mn_2 d_1)}{md_1 d_2} =_{def\ mdc} \frac{m \times mdc(n_1 d_2, n_2 d_1)}{m \times d_1 d_2} = mdc_g\left(\frac{n_1}{d_1}, \frac{n_2}{d_2}\right).$$

□

Verifica-se que a aplicação máximo divisor comum generalizado tem propriedades semelhantes às da aplicação máximo divisor comum dos naturais.

Proposição 2.2.9 Sejam $\delta_1, \delta_2 \in \Delta$ dois valores de tempo. A aplicação mdc_g determina o maior racional q tal que existem naturais m_1, m_2 tais que $q \times m_1 = \delta_1$ e $q \times m_2 = \delta_2$.

Demonstração:
Sejam $n_1, d_1, n_2, d_2 \in I\!N$ tais que

$$\delta_1 = \frac{n_1}{d_1} \text{ e } \delta_2 = \frac{n_2}{d_2}.$$

O resultado calculado por $mdc_g(\delta_1, \delta_2)$ é:

$$mdc_g(\delta_1, \delta_2) = mdc_g\left(\frac{n_1}{d_1}, \frac{n_2}{d_2}\right) = \frac{mdc(n_1 d_2, n_2 d_1)}{d_1 d_2} = \frac{b}{d_1 d_2} = q,$$

onde $b = mdc(n_1 d_2, n_2 d_1)$. Por definição de máximo divisor comum de dois naturais, existem naturais m_1 e m_2 tais que

$$b \times m_1 = n_1 d_2 \text{ e } b \times m_2 = n_2 d_1.$$

Ao multiplicar estes naturais por $mdc_g(\delta_1, \delta_2)$ obtemos as igualdades:

$$q \times m_1 = \frac{b}{d_1 d_2} \times m_1 = \frac{b \times m_1}{d_1 d_2} = \frac{n_1 d_2}{d_1 d_2} = \delta_1,$$

$$q \times m_2 = \frac{b}{d_1 d_2} \times m_2 = \frac{b \times m_2}{d_1 d_2} = \frac{n_2 d_1}{d_1 d_2} = \delta_2.$$

Para verificar que o racional q é o maior racional que satisfaz a duas igualdades anteriores, supomos que existem um racional q' e dois naturais p_1, p_2 tais que $q' \times p_1 = \delta_1$ e $q' \times p_2 = \delta_2$. Temos então que:

$$\begin{cases} q' \times p_1 = \delta_1 \\ q' \times p_2 = \delta_2 \end{cases} \leftrightarrow \begin{cases} q' \times p_1 = \frac{n_1}{d_1} \\ q' \times p_2 = \frac{n_2}{d_2} \end{cases} \leftrightarrow \begin{cases} d_1 d_2 q' \times p_1 = n_1 d_2 \\ d_1 d_2 q' \times p_2 = n_2 d_1 \end{cases}$$

Isto é, o número $d_1 d_2 q'$ é simultaneamente divisor de $n_1 d_2$ e de $n_2 d_1$. Por definição de máximo divisor comum podemos concluir que:

$$d_1 d_2 q' \leq b \leftrightarrow q' \leq \frac{b}{d_1 d_2} \leftrightarrow q' \leq q.$$

□

Ao calcular o máximo divisor comum generalizado entre dois naturais, obtemos o seu máximo divisor comum natural. Apresentamos alguns exemplos:

Exemplo 2.2.10 Alguns casos simples de aplicação de mdc_g:

- $mdc_g(6, 4) = 2$,
- $mdc_g(0.4, 1) = 0.2$,
- $mdc_g(0.66(6), 1.16(6)) = 0.16(6)$.

□

Para calcular o valor do máximo divisor comum generalizado entre dois racionais é necessário decompô-los recorrendo à divisão de naturais. Como a decomposição de racionais nem sempre é trivial, será apresentada uma definição alternativa para o máximo divisor comum generalizado que permite calcular o seu valor sem ser necessário recorrer à decomposição dos racionais.

Proposição 2.2.11 (Máximo Divisor Comum Generalizado*)
A seguinte definição de máximo divisor comum generalizado

$$mdc_g^*(\delta_1, \delta_2) = \begin{cases} \delta_2 & \text{se } \delta_1 = \delta_2 \text{ ou } \delta_1 = \iota \\ mdc_g^*(\delta_1 -_\iota \delta_2, \delta_2) & \text{se } \delta_1 \neq \iota \text{ e } \delta_2 <_\iota \delta_1 \\ mdc_g^*(\delta_2, \delta_1) & \text{se } \delta_1 <_\iota \delta_2 \end{cases}$$

é equivalente à definição anterior.

Demonstração:
Sejam $\delta_1, \delta_2 \in \Delta_\iota$. Consideremos três casos:

- $\delta_1 = \iota$ ou $\delta_1 = \delta_2$
  Temos que
  $$mdc_g^*(\delta_1, \delta_2) = \delta_2 = mdc_g(\iota, \delta_2),$$
- $\delta_2 = \iota$
  Temos que
  $$mdc_g^*(\delta_1, \delta_2) = mdc_g^*(\iota, \delta_1) = \delta_1 = mdc_g(\delta_1, \iota),$$
- $\delta_1 = \frac{n_1}{d_1}$ e $\delta_2 = \frac{n_2}{d_2}$
  Para este caso estabelecemos as seguintes igualdades:
  $$mdc_g^*(\delta_1, \delta_2) = mdc_g^*\left(\frac{n_1}{d_1}, \frac{n_2}{d_2}\right) = mdc_g^*\left(\frac{n_1 d_2}{d_1 d_2}, \frac{n_2 d_1}{d_1 d_2}\right).$$

  Uma vez que os dois operadores são racionais com o mesmo denominador, a subtracção do maior pelo menor consiste numa subtração entre os numeradores, que são naturais, mantendo-se o denominador. Esta técnica é semelhante à técnica utilizada para calcular iterativamente o máximo divisor comum entre dois naturais. Podemos então estabelecer as seguintes igualdades:
  $$mdc_g^*\left(\frac{n_1 d_2}{d_1 d_2}, \frac{n_2 d_1}{d_1 d_2}\right) = \frac{mdc(n_1 d_2, n_2 d_1)}{d_1 d_2} = mdc_g(\delta_1, \delta_2).$$

Podemos concluir que a aplicação mdc_g^* é idêntica à função mdc_g.

□

Capítulo 3

Lógica de Objectos: Sintaxe e Semântica

3.1 Introdução

É usual introduzir tipos de dados abstractos para permitir a definição de atributos e de acções com parâmetros, [Sernadas e Sernadas 94, Sernadas *et al* 95]. No entanto, esta opção obriga a que se recorra à lógica de primeira ordem para exprimir as propriedades dos objectos. Esta lógica é correcta e completa, mas é semidecidível, isto é, se uma fórmula não é válida então pode ser impossível, em tempo finito, verificar que a fórmula não é teorema. Para obter a decidibilidade é necessário recorrer à lógica proposicional. Porém, esta lógica é incompatível com tipos de dados abstractos, uma vez que os seus domínios poderão ser infinitos. Uma solução consistiria em restringir os atributos a expressões booleanas e em eliminar as acções parametrizáveis. Uma outra solução, que iremos adoptar, seria a de trabalhar apenas com domínios de cardinalidade finita nos atributos e nas acções. Esta última, ao contrário da primeira, permite alguma expressibilidade, embora limitada, possibilitando ainda a utilização de variáveis nos atributos e nas acções, o que facilita a construção das propriedades.

Definição 3.1.1 (Domínio de Valores)
Um domínio de valores $\mathcal{D}$ é um conjunto cujos elementos são conjuntos finitos e que contém conjunto vazio.

□

Exemplo 3.1.2 O conjunto $\mathcal{D}$ contém, por exemplo, os seguintes conjuntos:

- $\{verd, falso\}$,
- $\{esq, dir\}$,

- $\mathcal{N}_n = \{0, \ldots, n\}$ para qualquer $n \in I\!N_o$.

□

A imposição de cardinalidade finita é feita apenas ao nível dos conjuntos em $\mathcal{D}$, sendo arbitrário o número desses conjuntos.

As variáveis constituem também uma família de conjuntos finitos. Cada conjunto de variáveis está associado a um elemento da família $\mathcal{D}$. Porém, nem todos os elementos de $\mathcal{D}$ contêm um conjunto de variáveis. Não faz sentido falar de variáveis para um domínio quando esse domínio é vazio. Assim, a família de variáveis é definida do seguinte modo:

Definição 3.1.3 (Família de Variáveis)
A família $X = \{X_D\}_{D \in \mathcal{D}}$ de conjuntos finitos diz-se uma família de variáveis se verificar as seguintes propriedades:

- $X_\emptyset = \emptyset$,
- $X_D \cap X_{D'} = \emptyset$ para quaisquer $D, D' \in \mathcal{D}$ distintos,
- $X_D \cap D' = \emptyset$ para quaisquer $D, D' \in \mathcal{D}$.

□

3.2 Sintaxe de Objectos

Um objecto é caracterizado por executar acções, conter atributos e apresentar um comportamento, [Sernadas e Sernadas 94]. Na assinatura de um objecto são indicadas as acções e os atributos, bem como o intervalo de tempo a partir do qual são obtidos os valores para o relógio do objecto. O comportamento do objecto será mais tarde indicado utilizando fórmulas de uma lógica apropriada.

3.2.1 Assinatura e Termos

Fixamos um domínio de valores $\mathcal{D}$ e uma família de variáveis X.

Definição 3.2.1 (Assinatura de um Objecto)
Uma assinatura de objecto é um quádruplo $\Sigma_{ob} = \langle \Delta_{ob}, \tau_{ob}, GACT_{ob}, GATT_{ob} \rangle$, onde

- Δ_{ob} é um conjunto de valores de tempo,

- $\tau_{ob} \in (\Delta_{ob})^+_\iota$,
- $GACT_{ob}$ é um triplo $\langle ACT_{ob}, par_{ob}, dur_{ob}\rangle$ em que
  - ACT_{ob} é um conjunto finito que contém $\perp$ e cujos elementos se dizem acções,
  - $par_{ob} : ACT_{ob} \to \mathcal{D}$ é uma aplicação tal que $par_{ob}(\perp) = \emptyset$,
  - $dur_{ob} : ACT_{ob} \to \Delta_{ob} \cup \{0\}$ é uma aplicação tal que $dur_{ob}(\perp) = 0$;

Para definir o elemento $GATT_{ob}$ é necessário considerar os seguintes conjuntos:

$$\kappa_{ob} = \{act \ : \ act \in ACT_{ob} \text{ e } par_{ob}(act) = \emptyset\} \cup$$

$$\{act(d) \ : \ act \in ACT_{ob} \text{ e } d \in par_{ob}(act)\},$$

$$\mathcal{D}_{ob} = \mathcal{D} \cup \{A \subseteq \kappa_{ob} : \text{ se } act(d) \in A, \text{ então para todo o } d' \in par_{ob}(act),$$

$$act(d') \in A\};$$

- $GATT_{ob}$ é um triplo $\langle ATT_{ob}, res_{ob}, ANM_{ob}\rangle$ em que
  - ATT_{ob} é um conjunto finito cujos elementos se dizem atributos,
  - $res_{ob} : ATT_{ob} \to \mathcal{D}_{ob}$ é uma aplicação onde $res_{ob}(att) \neq \emptyset$ para qualquer $att \in ATT_{ob}$,
  - ANM_{ob} é um subconjunto de $\{(att_1, att_2) : \ att_1, att_2 \in ATT \text{ e } att_1 \neq att_2\}$ onde:
    * para qualquer atributo att existe no máximo um par $(att_1, att_2) \in ANM_{ob}$ em que $att = att_1$ ou $att = att_2$,
    * para qualquer par $(att_1, att_2) \in ANM_{ob}$, $res_{ob}(att_1) = res_{ob}(att_2) \subseteq \kappa_{ob}$.

Uma assinatura de objecto diz-se uma *assinatura de objecto primário* se $occ_i, occ_f \in ATT_{ob}$, $ANM_{ob} = \{(occ_i, occ_f)\}$ e $r_{ob}(occ_i) = r_{ob}(occ_f) = \kappa_{ob}$.

□

Os quatro elementos que constituem uma assinatura de objecto são os seguintes: um conjunto de domínio de tempo, um valor do domínio de tempo, um gerador de acções e um gerador de atributos.

O conjunto domínio de tempo poderá ser qualquer conjunto desde o conjunto vazio, no caso da lógica temporal pura, até ao conjunto de todos os racionais positivos. É através desse conjunto que são indicados os valores de tempo que o relógio do objecto poderá medir. Por exemplo, no caso de o conjunto ser limitado por um racional q, isto quer dizer que o relógio do objecto em causa não tem a capacidade de medir o tempo para valores superiores a q, valores esses que são interpretados como sendo ι (indefinido). Portanto, não se faz a distinção entre valores de tempo superiores a q. Consideremos alguns casos práticos: um relógio de pulso tem uma capacidade limite de doze horas; uma base de dados poderá guardar informações com menos de dois anos, sendo a sua capacidade limitada por dois anos. Ainda neste capítulo, será apresentada uma métrica para calcular a *distância em tempo* entre dois eventos. A definição dessa métrica irá depender do limite superior atribuído ao conjunto de valores de tempo de uma assinatura.

O segundo elemento da assinatura consiste no período do relógio do objecto, τ_{ob}. O valor do período não poderá ser nulo, mas poderá ser indefinido (ι). Esta última situação verifica-se, por exemplo, no caso da lógica temporal pura.

O gerador de acções é responsável pela definição das acções de um de objecto. É constituído por três componentes: o conjunto cujos elementos são intitulados acções, uma aplicação que associa um elemento do domínio de valores a uma acção, e uma aplicação que associa um valor de tempo a uma acção. No conjunto das acções encontra-se o elemento $\perp$. Na lógica temporal para raciocínio de objectos introduzida em [Sernadas e Sernadas 94, Sernadas *et al* 95], $\perp$ representa a acção de inércia e é utilizado para indicar que o objecto não está a executar nenhuma outra acção. Deste modo, o valor dos atributos do objecto não é alterado sempre que $\perp$ seja a acção a ser executada pelo objecto. Com a introdução do tempo, estas duas propriedades de $\perp$ deixam de ser verificadas. Desde logo, $\perp$ deixa de ter um comportamento que é característico das restantes acções da assinatura e, consequentemente, não é possível associar-lhe a imutabilidade dos atributos. No entanto, o elemento $\perp$ é inserido no conjunto ACT_{ob} por razões práticas. Através da aplicação par_{ob} são indicados os valores que as acções podem ter como parâmetros. Para facilitar futuros raciocínios, só é permitido no máximo um parâmetro para cada acção. As acções com mais do que um parâmetro podem ser especificadas se considerarmos domínios formados por tuplos de outros domínios. Uma vez que os elementos que constituem os tuplos são finitos, então o conjunto formado pelos tuplos também é finito. Poderão existir acções sem parâmetros; por exemplo, a acção $\perp$. Por fim, a aplicação dur_{ob} indica o tempo de execução de uma acção. Este tempo poderá ser positivo ou nulo. No caso de ser nulo diz-se que a acção é atómica. No âmbito da lógica temporal pura, onde o domínio de tempo é vazio, todas as acções são atómicas. O elemento $\perp$ apresenta uma duração de execução nula. A única razão por detrás desta opção é possibilidade de definir dur_{ob} como uma aplicação, e não como uma função parcial.

Para concluir a definição da assinatura, é necessário enriquecer o domínio de valores com novos conjuntos que contêm acções. Começamos por construir o conjunto das acções instanciadas com todos os valores do seu parâmetro. Este conjunto é designado por κ_{ob}. Podemos concluir que κ_{ob} é finito uma vez que o conjunto das acções é finito, bem como o parâmetro de cada acção. Do conjunto κ_{ob} serão retirados subconjuntos que são acrescentados ao domínio de valores. No entanto, nem todo o subconjunto de κ_{ob} será incluído no domínio de valores. Para que haja coerência, iremos considerar apenas os subconjuntos que contenham todas as possíveis instanciações das acções nele contidas. Isto é, ou o conjunto não contém nenhuma instanciação de uma certa acção, ou contém todas as suas instanciações. O novo domínio de valores é representado por $\mathcal{D}_{ob}$. Para simplificar futuros resultados, consideremos a seguinte restrição sobre os conjuntos do domínio de valores: para qualquer conjunto D de $\mathcal{D}$, a intersecção desse conjunto com κ_{ob} é vazia. Isto é, nenhum conjunto do domínio de valores pode conter acções ou instanciações de acções.

No quarto elemento da definição da assinatura é indicada toda a informação relacionada com os atributos do objecto. O conjunto ATT_{ob} contém as designações dos atributos do objecto, sendo a aplicação res_{ob} responsável por atribuir um conjunto do domínio de valores a cada atributo. O conjunto dos atributos não poderá ser vazio, tendo de conter no mínimo dois elementos para definir o conjunto ANM_{ob}. Este último conjunto é responsável pela animação do objecto. Na literatura, existem duas técnicas para indicar a animação de um objecto: através de predicados unários cujo domínio seja acções [Sernadas *et al* 95], ou recorrendo a um atributo especial [Sernadas e Sernadas 94]. Nesta dissertação optamos pela segunda técnica. No entanto, como existe a possibilidade de definir acções não atómicas, é necessário considerar pares de atributos para representar a animação. O conjunto ANM_{ob} é formado por esses pares de atributos. O primeiro elemento do par indica o início da execução de uma acção, sendo o seu fim indicado posteriormente através do segundo elemento.

Das possíveis assinaturas de objectos, destaca-se a assinatura de objecto primário. Esta assinatura é caracterizada por conter um único par de atributos de animação. Esses atributos são designados por occ_i e occ_f, podendo tomar qualquer valor em κ_{ob}.

Notação 3.2.2 Estendemos a aplicação dur_{ob} de modo a aceitar elementos de κ_{ob}. Seja $act(d) \in \kappa_{ob}$, então $dur_{ob}(act(d)) = dur_{ob}(act)$.

□

No que se segue, e sempre que não haja ambiguidade, omitimos o índice *ob* na definição da assinatura.

Como exemplo de assinaturas de objectos, apresentamos uma versão simplificada do *distribuidor de pacotes* apresentada em [Böhm e Sernadas 93]. O sistema consiste numa distribuição de pacotes por uma rede de dispositivos binários. Esta rede é composta por quatro componentes distintos: o dispositivo binário que direcciona o pacote para uma das duas possíveis saídas; um tapete por onde são movimentados os pacotes através dos distribuidores binários; um dispositivo unário que se encontra nas extremidades da rede para receber os pacotes; e um sensor no início da rede para a leitura do destino do pacote. No início da rede, o sensor lê do pacote qual é o se destino. Esta leitura é depois comunicada ao primeiro dispositivo binário da rede para que ele decida qual a saída para a qual deverá direccionar o pacote. Este direccionamento é feito por meio de uma patilha que faz a obstrução da saída não desejada. A obstrução tem que ser feita de modo a não danificar os pacotes que passam pelo dispositivo. Em seguida, tendo já conhecimento de qual foi a saída do pacote, este dispositivo deverá comunicar ao seguinte que o pacote se encontra a caminho. E assim sucessivamente até que o pacote chegue a um dispositivo unário. No nosso exemplo, consideramos apenas uma parte da rede com três objectos: `pacote` que representará o pacote a ser distribuído; `dist_bin` que representará um dispositivo binário que contém uma patilha e direcciona os pacotes para uma das duas saídas possíveis e `tapete` que representará o tapete que efectua a deslocação do pacote ao longo da rede. Uma vez que não haverá sensor, o dispositivo binário irá obstruir alternadamente uma das duas saídas sem causar danos aos pacotes que por ele passam.

Exemplo 3.2.3 Supondo que existem no máximo $n \in I\!N$ pacotes para distribuir, as assinaturas dos três objectos primários são as seguintes:

1. `pacote`:
   - $\Delta_{pacote} = \emptyset$,
   - $\tau_{pacote} = \iota$,
   - $ACT_{pacote} = \{\texttt{obs\_id}, \bot, \star\}$ onde
     - $par_{pacote}(\texttt{obs\_id}) = \mathcal{N}_n$,
     - $par_{pacote}(\star) = \mathcal{N}_n$,
     - $dur_{pacote}(\texttt{obs\_id}) = 0$,
     - $dur_{pacote}(\star) = 0$,
   - $ATT_{pacote} = \{\texttt{id}, occ_i, occ_f\}$ onde
     - $res_{pacote}(\texttt{id}) = \mathcal{N}_n$,
     - $res_{pacote}(occ_i) = \kappa_{pacote}$,
     - $res_{pacote}(occ_f) = \kappa_{pacote}$,
   - $ANM_{pacote} = \{(occ_i, occ_f)\}$;
2. `tapete`:
   - $\Delta_{tapete} = [0, 6[\cap \mathbb{Q}$,
   - $\tau_{tapete} = 5$,
   - $ACT_{tapete} = \{\texttt{perc}, \bot, \star\}$ onde
     - $par_{tapete}(\texttt{perc}) = \mathcal{N}_n$,
     - $par_{tapete}(\star) = \emptyset$,
     - $dur_{tapete}(\texttt{perc}) = 5$,
     - $dur_{tapete}(\star) = 0$,
   - $ATT_{tapete} = \{occ_i, occ_f\}$ onde
     - $res_{tapete}(occ_i) = \kappa_{tapete}$,
     - $res_{tapete}(occ_f) = \kappa_{tapete}$,

- $ANM_{tapete} = \{(occ_i, occ_f)\}$;

3. `dist_bin`:

- $\Delta_{dist\_bin} = \mathbb{Q}_o^+$,
- $\tau_{dist\_bin} = 2$,
- $ACT_{dist\_bin} = \{\texttt{entr\_pct}, \texttt{saída\_pct}_1, \texttt{saída\_pct}_2, \texttt{md\_pat}, \perp, \star\}$ onde
  - $par_{dist\_bin}(\texttt{entr\_pct}) = \mathcal{N}_n$,
  - $par_{dist\_bin}(\texttt{saída\_pct}_1) = \mathcal{N}_n$,
  - $par_{dist\_bin}(\texttt{saída\_pct}_2) = \mathcal{N}_n$,
  - $par_{dist\_bin}(\texttt{md\_pat}) = \emptyset$,
  - $par_{dist\_bin}(\star) = \emptyset$,
  - $dur_{dist\_bin}(act) = 0$ para qualquer $act \in ACT_{dist\_bin}$,
- $ATT_{dist\_bin} = \{\texttt{direc}, occ_i, occ_f\}$ onde
  - $res_{dist\_bin}(\texttt{direc}) = \{esq, dir\}$,
  - $res_{dist\_bin}(occ_i) = \kappa_{dist\_bin}$,
  - $res_{dist\_bin}(occ_f) = \kappa_{dist\_bin}$,
- $ANM_{dist\_bin} = \{(occ_i, occ_f)\}$.

□

Dada uma assinatura, indicamos agora como são construídos os termos para essa assinatura. A técnica de construção é muito semelhante à utilizada no âmbito da lógica de primeira ordem com tipos de dados abstractos. No entanto, é necessário ter em conta que existem conjuntos do domínio de valores que contêm acções.

Definição 3.2.4 (Termos de uma Assinatura)
Os termos de uma assinatura Σ constituem uma família $T_\Sigma = \{T_{\Sigma,D}\}_{D\in\mathcal{D}}$ de conjuntos definidos indutivamente do seguinte modo:

- $d \in T_{\Sigma,D}$ se $d \in D$,
- $x \in T_{\Sigma,D}$ se $x \in X_D$,
- $att \in T_{\Sigma,res(att)}$ se $att \in ATT$,
- $act(t) \in T_{\Sigma,D}$ se $act(d) \in D$ e $t \in T_{\Sigma,par(act)}$ para $D \subseteq \kappa$.

□

O conjunto dos termos para um domínio D contém: os elementos desse domínio, as variáveis associadas ao domínio e os atributos que tomam valores no domínio. Para o caso particular de um subconjunto de κ, o conjunto dos termos contém ainda as acções instanciadas com termos do seu parâmetro. Quando o domínio é vazio, o conjunto dos seus termos também é vazio. O conjunto dos termos para qualquer $D \in \mathcal{D}$ é finito.

Para manipular os termos, é necessário definir duas aplicações: uma que indique as variáveis que ocorrem nos termos e outra que substitua variáveis por termos.

Definição 3.2.5 (Variáveis de um Termo)
Seja T_Σ a família dos termos de Σ. A aplicação var_Σ, que a cada termo faz corresponder o conjunto das variáveis que ocorrem nesse termo, é definida indutivamente do seguinte modo:

- $var_\Sigma(d) = \emptyset$ se $d \in D$ para algum $D \in \mathcal{D}$,
- $var_\Sigma(x) = \{x\}$ se $x \in X_D$ para algum $D \in \mathcal{D}$,
- $var_\Sigma(att) = \emptyset$ se $att \in ATT$,
- $var_\Sigma(act(t)) = var_\Sigma(t)$ se $act(t) \in T_{\Sigma,D}$.

□

Definição 3.2.6 (Substituição em Termos)
Sejam $\{x_1, \ldots, x_n\}$ um conjunto de variáveis e $\{t_1, \ldots, t_n\}$ um conjunto de termos tais que se $x_i \in X_D$ então $t_i \in T_{\Sigma,D}$. A substituição das variáveis $\{x_1, \ldots, x_n\}$ por $\{t_1, \ldots, t_n\}$, num termo, define-se indutivamente do seguinte modo:

- $[t]^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} = t$ se $x_i \notin var_\Sigma(t)$ para qualquer $i = 1, \ldots, n$,
- $[x_i]^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} = t_i$ se $1 \leq i \leq n$,
- $[act(x_i)]^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} = act(t_i)$ se $1 \leq i \leq n$,.

□

Os únicos termos que poderão conter variáveis são as próprias variáveis ou as acções instanciadas com uma variável do parâmetro. Na definição anterior, qualquer termo verifica um dos três casos.

3.2.2 Fórmulas

As fórmulas para a lógica são agrupadas em dois conjuntos: o conjunto das fórmulas atómicas e o conjunto das fórmulas temporais. O primeiro contém as fórmulas mais simples da lógica, as fórmulas que não podem ser subdivididas em outras fórmulas. O segundo é construído a partir das fórmulas atómicas utilizando os conectivos lógicos $\Rightarrow, \neg, \forall$, e os operadores temporais. Alguns desses operadores, X, Y, pertencem à lógica temporal pura [Fitting 83] ao passo que outros são enriquecimentos dos operadores temporais U, S com valores de tempo, [Koymans 89, Koymans 92]. Os restantes operadores da lógica temporal pura apresentados em [Fitting 83] são abreviaturas dos anteriores.

Definição 3.2.7 (Fórmulas Atómicas)
O conjunto das fórmulas atómicas para a assinatura Σ é definido do seguinte modo:

$$A_\Sigma = \{(t = t') : \ t, t' \in T_{\Sigma,D} \text{ e } D \in \mathcal{D}\}$$

□

Uma fórmula diz-se atómica se corresponde a uma igualdade entre dois termos. Não existe garantia de que o conjunto das fórmulas atómicas seja finito, uma vez que o número de domínios poderá ser infinito. Este resultado não irá comprometer a decidibilidade da lógica. Uma vez que os termos não constituem uma família vazia, o conjunto das fórmulas atómicas também não é vazio.

Definição 3.2.8 (Fórmulas Temporais)
O conjunto das fórmulas temporais para a assinatura Σ com conjunto de valores de tempo Δ, F_Σ, é definido indutivamente do seguinte modo:

- $p \in F_\Sigma$

  se $p \in A_\Sigma$,

- $(\neg\, \varphi), (\mathsf{X}\, \varphi), (\mathsf{Y}\, \varphi), (\forall x\ \varphi) \in F_\Sigma$

  se $\varphi \in F_\Sigma$ e $x \in X_D$ para algum $D \in \mathcal{D}$,

- $(\varphi_1 \Rightarrow \varphi_2), (\varphi_1\, \mathsf{U}_\delta\, \varphi_2), (\varphi_1\, \mathsf{S}_\delta\, \varphi_2) \in F_\Sigma$

  se $\varphi_1, \varphi_2 \in F_\Sigma$ e $\delta \in \Delta$,

- $(\varphi_1\, \mathsf{U}_{\preceq\delta}\, \varphi_2), (\varphi_1\, \mathsf{S}_{\preceq\delta}\, \varphi_2) \in F_\Sigma$

  se $\varphi_1, \varphi_2 \in F_\Sigma$ e $\delta \in \Delta_\iota$.

□

As fórmulas $(\neg\,\varphi)$, $(\varphi_1 \Rightarrow \varphi_2)$, $(\mathsf{X}\,\varphi)$ e $(\mathsf{Y}\,\varphi)$ têm nesta lógica a mesma interpretação que têm na lógica temporal pura apresentada em [Goldblatt 87, Fitting 83]. A fórmula $(\forall x\ \varphi)$ também é interpretada de modo semelhante, embora, como os domínios são finitos, seja possível visualizá-la como sendo uma conjunção finita de substituições da variável x pelos elementos do seu domínio em φ. Apesar desta semelhança, não é possível, a nível sintáctico, introduzir esta fórmula como abreviatura das restantes. Como as conjunções são binárias, uma conjunção entre três elementos pode tomar seis aspectos diferentes, o que corresponde a seis fórmulas distintas do ponto de vista sintáctico.

Nas restantes fórmulas já se entra em conta com valores de tempo. A fórmula $(\varphi_1\,\mathsf{U}_\delta\,\varphi_2)$ tem a seguinte interpretação: verifica-se φ_1 até que seja verificado φ_2 à distância de δ unidades de tempo no futuro. A fórmula $(\varphi_1\ \mathsf{S}_\delta\ \varphi_2)$ tem o mesmo significado, mas no passado. O valor de δ nunca poderá ser ι, uma vez que não há a garantia da existência de um único momento no futuro (ou no passado) a distância ι do presente.

A fórmula $(\varphi_1\ \mathsf{U}_{\preceq\delta}\ \varphi_2)$ é interpretada de modo semelhante à anterior, com excepção do seguinte: a fórmula φ_2 poderá ser verificada numa distância inferior ou igual a δ no futuro, inclusivamente a distância zero, o que corresponde ao presente. Nestas fórmulas permite-se que δ seja ι, sendo a sua interpretação neste caso semelhante à interpretação da fórmula na lógica temporal pura $(\varphi_1\ \mathsf{U}\ \varphi_2)$. De modo análogo se interpreta a fórmula $(\varphi_1\,\mathsf{S}_{\preceq\delta}\,\varphi_2)$.

Notação 3.2.9 (Abreviaturas)
Definimos agora algumas abreviaturas que irão ser úteis para futuros raciocínios:

- $(tt) \equiv_{abv} (t = t)$,
- $(\varphi_1 \vee \varphi_2) \equiv_{abv} ((\neg\,\varphi_1) \Rightarrow \varphi_2)$,
- $(\varphi_1 \wedge \varphi_2) \equiv_{abv} (\neg(\varphi \Rightarrow (\neg\,\varphi_2)))$,
- $(\varphi_1 \Leftrightarrow \varphi_2) \equiv_{abv} ((\varphi \Rightarrow \varphi_2) \wedge (\psi \Rightarrow \varphi))$,
- $(\exists x\ \varphi) \equiv_{abv} (\neg(\forall x\ (\neg\,\varphi)))$,
- $(\varphi_1\ \mathsf{U}\ \varphi_2) \equiv_{abv} (\varphi_1\ \mathsf{U}_{\preceq\iota}\,\varphi_2)$,
- $(\mathsf{F}\,\varphi) \equiv_{abv} (tt\ \mathsf{U}\ \varphi)$,
- $(\mathsf{F}_\delta\,\varphi) \equiv_{abv} (tt\ \mathsf{U}_\delta\,\varphi)$,
- $(\mathsf{F}_{\preceq\delta}\,\varphi) \equiv_{abv} (tt\ \mathsf{U}_{\preceq\delta}\,\varphi)$,
- $(\mathsf{G}\,\varphi) \equiv_{abv} (\neg(\mathsf{F}(\neg\,\varphi)))$,
- $(\mathsf{G}_\delta\,\varphi) \equiv_{abv} (\neg(\mathsf{F}_\delta(\neg\,\varphi)))$,
- $(\mathsf{G}_{\preceq\delta}\,\varphi) \equiv_{abv} (\neg(\mathsf{F}_{\preceq\delta}(\neg\,\varphi)))$,

- $(\varphi_1\,\mathsf{S}\,\varphi_2) \equiv_{abv} (\varphi_1\,\mathsf{S}_{\preceq\iota}\,\varphi_2)$,
- $(\mathsf{P}\,\varphi) \equiv_{abv} (tt\,\mathsf{S}\,\varphi)$,
- $(\mathsf{P}_\delta\,\varphi) \equiv_{abv} (tt\,\mathsf{S}_\delta\,\varphi)$,
- $(\mathsf{P}_{\preceq\delta}\,\varphi) \equiv_{abv} (tt\,\mathsf{S}_{\preceq\delta}\,\varphi)$,
- $(\mathsf{H}\,\varphi) \equiv_{abv} (\neg(\mathsf{P}(\neg\,\varphi)))$,
- $(\mathsf{H}_\delta\,\varphi) \equiv_{abv} (\neg(\mathsf{P}_\delta(\neg\,\varphi)))$,
- $(\mathsf{H}_{\preceq\delta}\,\varphi) \equiv_{abv} (\neg(\mathsf{P}_{\preceq\delta}(\neg\,\varphi)))$.

□

As fórmulas temporais $(\varphi_1\,\mathsf{U}\,\varphi_2)$ e $(\varphi_1\,\mathsf{S}\,\varphi_2)$ são definidas como abreviaturas utilizando os operadores $\mathsf{U}_{\preceq\iota}$ e $\mathsf{S}_{\preceq\iota}$. As restantes fórmulas são definidas do modo usual para a lógica temporal pura. Novamente recordamos que o futuro e o passado contêm o presente, o que implica que $(\varphi_1\,\mathsf{U}\,\varphi_2)$ e $(\varphi_1\,\mathsf{S}\,\varphi_2)$ são verificadas se φ_2 o for.

Para completar a definição de uma especificação de objecto, é necessário impor restrições ao comportamento do mesmo. As restrições podem ser impostas a nível sintáctico, através de fórmulas temporais, ou a nível semântico, através de propriedades que as estruturas de interpretação terão que satisfazer. As propriedades de animação e os efeitos das acções sobre os atributos são introduzidos a nível sintáctico. As restrições ao nível do comportamento das acções são impostas a nível semântico.

Definição 3.2.10 (Especificação de Objecto)
Uma especificação de objecto Ob é um par $\langle \Sigma_{ob}, Ax_{ob}\rangle$ constituído por uma assinatura de objecto e por um subconjunto finito de fórmulas de F_Σ. Os elementos do conjunto Ax dizem-se axiomas do objecto Ob.

□

Seguem-se as especificações de objectos com base nas assinaturas apresentadas no exemplo 3.2.3.

Exemplo 3.2.11 O conjunto dos axiomas para as assinaturas em 3.2.3 são os seguintes:

`pacote` o conjunto Ax_{pacote} contém as seguintes fórmulas

pt1. $(\mathsf{G}((occ_i = \star(x)) \Rightarrow ((\mathsf{X}(\mathsf{G}(\mathtt{id} = x))) \wedge (\neg(\mathsf{Y}\,tt)))))$,

pt2. $(\mathsf{G}((occ_i = \mathtt{obs\_id}(x)) \Rightarrow (\mathtt{id} = x)))$;

`tapete` $Ax_{tapete} = \{((occ_i = \star) \Rightarrow (\neg(\mathsf{Y}\,tt)))\}$

`dist_bin` Os axiomas de `dist_bin`, Ax_{dist_bin}, são os seguintes:

db1. $(\mathsf{G}((occ_i = \star) \Rightarrow ((\mathsf{X}(\mathtt{direc} = left)) \wedge (\neg(\mathsf{Y}\,tt)))))$,

db2. $(\mathsf{G}((occ_i = \mathtt{entr\_pct}(x)) \Rightarrow (\mathsf{F}((occ_i = \mathtt{saída\_pct}_1(x)) \vee$
$(occ_i = \mathtt{saída\_pct}_2(x))))))$,

db3. $(\mathsf{G}((occ_i = \mathtt{saída\_pct}_1(x)) \Rightarrow (\mathsf{P}(occ_i = \mathtt{entr\_pct}(x)))))$,

db4. $(\mathsf{G}((occ_i = \mathtt{saída\_pct}_2(x)) \Rightarrow (\mathsf{P}(occ_i = \mathtt{entr\_pct}(x)))))$,

db5. $(\mathsf{G}(((occ_i = \mathtt{saída\_pct}_1(x)) \vee (occ_i = \mathtt{saída\_pct}_2(x))) \Rightarrow (\mathsf{X}(occ_i = \mathtt{md\_pat}))))$,

db6. $(\mathsf{G}((occ_i = \mathtt{md\_pat}) \Rightarrow (\mathsf{Y}((occ_i = \mathtt{saída\_pct}_1(x)) \vee (occ_i = \mathtt{saída\_pct}_2(x))))))$,

db7. $(\mathsf{G}(((occ_f = \mathtt{md\_pat}) \wedge (\mathtt{direc} = left)) \Rightarrow (\mathsf{X}(\mathtt{direc} = right))))$,

db8. $(\mathsf{G}(((occ_f = \mathtt{md\_pat}) \wedge (\mathtt{direc} = right)) \Rightarrow (\mathsf{X}(\mathtt{direc} = left))))$,

db9. $(\mathsf{G}((\mathtt{direc} = left) \Leftrightarrow (\neg(\mathtt{direc} = right))))$.

□

Como na secção anterior, estamos interessados em manipular sintacticamente as fórmulas. Essa manipulação só irá ser permitida ao nível das variáveis, através da substituição destas por termos. Em seguida apresentamos resultados essenciais para a manipulação das fórmulas.

Definição 3.2.12 (Variáveis que Ocorrem Livres numa Fórmula)
Sejam Σ uma assinatura e F_Σ o conjunto das fórmulas de Σ. A aplicação vl_Σ que a cada fórmula associa o conjunto das variáveis que ocorrem livres é definida indutivamente do seguinte modo:

- $vl_\Sigma(t = t') = var_\Sigma(t) \cup var_\Sigma(t')$,
- $vl_\Sigma(\mathsf{N}\varphi) = vl_\Sigma(\varphi)$ para $\mathsf{N} \in \{\neg, \mathsf{X}, \mathsf{Y}\}$,
- $vl_\Sigma(\forall x\ \varphi) = vl_\Sigma(\varphi) - \{x\}$,
- $vl_\Sigma(\varphi_2 \mathsf{B} \varphi_2) = vl_\Sigma(\varphi_1) \cup vl_\Sigma(\varphi_2)$ para $\mathsf{B} \in \{\Rightarrow, \mathsf{U}_\delta, \mathsf{U}_{\preceq\delta}, \mathsf{S}_\delta, \mathsf{S}_{\preceq\delta}\}$.

□

Definição 3.2.13 (Substituição em Fórmulas)
Sejam $\{x_1, \ldots, x_n\}$ um conjunto de variáveis e $\{d_1, \ldots, d_n\}$ um conjunto de elementos tal que se $x_i \in X_D$ então $d_i \in D$. A substituição das variáveis $\{x_1, \ldots, x_n\}$ pelos elementos $\{d_1, \ldots, d_n\}$ em fórmulas define-se indutivamente do seguinte modo:

- $[(t = t')]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} = \left([t]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} = [t']^{x_1,\ldots,x_n}_{d_1,\ldots,d_n}\right)$,
- $[(\mathsf{N}\varphi)]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} = \left(\mathsf{N}[\varphi]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n}\right)$ se $\mathsf{N} \in \{\neg, \mathsf{X}, \mathsf{Y}\}$,
- $[(\forall x\ \varphi)]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} = \left(\forall x\ [\varphi]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n}\right)$ se $x \notin \{x_1, \ldots, x_n\}$,
- $[(\forall x_i\ \varphi)]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} = \left(\forall x_i\ [\varphi]^{x'_1,\ldots,x'_{n-1}}_{d'_1,\ldots,d'_{n-1}}\right)$ para $1 \leq i \leq n$ onde

$$x'_j = \begin{cases} x_j & \text{se } j < i \\ x_{j+1} & \text{se } i \leq j < n \end{cases}$$

$$d'_j = \begin{cases} d_j & \text{se } j < i \\ d_{j+1} & \text{se } i \leq j < n \end{cases}$$

- $[(\varphi_1 \mathsf{B} \varphi_2)]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} = \left([\varphi_1]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n} \mathsf{B} [\varphi_2]^{x_1,\ldots,x_n}_{d_1,\ldots,d_n}\right)$ se $\mathsf{B} \in \{\Rightarrow, \mathsf{U}_\delta, \mathsf{U}_{\preceq\delta}, \mathsf{S}_\delta, \mathsf{S}_{\preceq\delta}\}$.

$\square$

A substituição sintáctica de uma variável por um termo do mesmo domínio numa fórmula será permitida se essa variável ocorrer livre na fórmula. As variáveis que não ocorrem livres são as variáveis quantificadas e a sua substituição por um termo poderá alterar a interpretação da fórmula inicial.

3.3 Semântica

3.3.1 Estrutura de Interpretação

Tal como na lógica temporal pura, recorremos às estruturas de Kripke para introduzir semântica nas fórmulas temporais. No entanto, ao trabalhar com tempo explícito e, consequentemente, com acções que apresentam uma duração de execução, é necessário enriquecer a estrutura de Kripke de modo a respeitar essas durações. A linearidade do tempo e a inexistência de ciclos são características dos objectos expressas através da imposição de propriedades na estrutura de Kripke.

Definição 3.3.1 (Estrutura de Kripke)
Uma estrutura de Kripke para uma assinatura Σ é um triplo $K = \langle W, R, V\rangle$ em que:

- W é um conjunto cujos elementos se dizem estados do objecto,
- $R \subseteq W \times W$ é uma relação, dita de visibilidade directa, com as seguintes propriedades:

- não existe nenhum conjunto de estados $\{w_i : i \in I\!N_o\}$ tal que $(w_{i+1}, w_i) \in R$ para todo $i \in I\!N_o$,
- não existem $w, w_0, \ldots, w_n \in W$ tais que
  * $w_0 = w, w_n = w$,

    e
  * $\{(w_0, w_1), \ldots, (w_{n-1}, w_n)\} \subset R$,
- se $(w, w_1), (w, w_2) \in R$ então $w_1 = w_2$,
- se $(w_1, w), (w_2, w) \in R$ então $w_1 = w_2$,

• $V = \{V_{att} : W \nrightarrow res(att)\}_{att \in ATT}$ é uma família de funções parciais, dita de valoração.

□

Contrariamente ao que é habitual numa estrutura de Kripke, a família de valoração é constituída por funções parciais etiquetadas por atributos. A parcialidade irá ser necessária para obter os resultados de correcção e completude do sistema dedutivo (4.4.1) apresentados no capítulo 4.

A relação R corresponde à visibilidade directa entre estados. Essa relação é linear à esquerda, linear à direita e acíclica. A primeira condição imposta sobre a relação de visibilidade garante a existência de um estado inicial para qualquer sequência de estados na relação R. Ao fazer o seu fecho reflexivo e transitivo obtemos a relação de visibilidade R^*.

Notação 3.3.2 Seja $R \subseteq W \times W$ uma relação de visibilidade directa na estrutura de Kripke K e sejam $w, w' \in W$.

• Diz-se que $(w, w') \in R^n$ para $n \in I\!N_o$ sse existem estados $w_0, \ldots, w_n \in W$ tais que:
  - $w_0 = w, w_n = w'$,

    e
  - $\{(w_0, w_1), \ldots, (w_{n-1}, w_n)\} \subseteq R$

• Diz-se que $(w, w') \in R^*$ sse existe $n \in I\!N_o$ tal que $(w, w') \in R^n$.

□

A relação de visibilidade directa não só representa explicitamente a evolução de um objecto ao longo do tempo, como representa também implicitamente o próprio tempo. Para uma assinatura Σ com período τ, uma mudança significativa dos seus atributos só se verifica em intervalos de τ unidades de tempo. Consequentemente, é desnecessário considerar estados que diferem no tempo de um valor inferior a τ. Optamos por considerar que dois estados se encontram na relação R se a diferença em tempo entre eles é de τ. Com este pressuposto, é possível definir uma função parcial que calcule o valor que separa dois estados no tempo. Esta função só se encontra definida para estados que se encontram na relação R^*.

Prop/Definição 3.3.3 (Métrica de uma Estrutura de Kripke)
Seja $K = \langle W, R, V\rangle$ uma estrutura de Kripke para a assinatura Σ com período τ. A função parcial $m_K : W \otimes W \nrightarrow \Delta_\iota \cup \{0\}$, definida indutivamente do seguinte modo

$$m_K(w_1, w_2) = \begin{cases} 0 & \text{se } w_1 = w_2 \\ \tau +_\iota m_K(w, w_2) & \text{se } (w_1, w) \in R \text{ e } (w, w_2) \in R^* \\ m_K(w_2, w_1) & \text{se } (w_2, w_1) \in R^* \\ \text{indefinido} & \text{caso contrário,} \end{cases}$$

encontra-se bem definida e estabelece uma métrica sobre os elementos de R^*.

Demonstração:
Sejam w, w' dois estados de W que se encontram na relação R^*. Por definição da relação de visibilidade, existem estados $w_0, \ldots, w_n$ tais que $\{(w_0, w_1), \ldots, (w_{n-1}, w_n)\} \subseteq R$ e, ou $w_0 = w$ e $w_n = w'$, ou $w_0 = w'$ e $w_n = w$. Calcular o valor de $m_K(w, w')$ consiste numa adição de um número finito de parcelas iguais à constante τ. Uma vez que a operação $+_\iota$ se encontra bem definida, a função m_K também se encontra bem definida.

□

Observação 3.3.4 Podemos distinguir três tipos de métricas, consoante o tipo de conjunto de valores de tempo e o valor de τ:

- $\tau = \iota$

  Neste caso o valor da função m_K para quaisquer $w, w' \in W$ na relação R^* poderá ser zero se $w = w'$, ou ι caso contrário.

- $\Delta = [0, q[\cap \mathbb{Q}$ para um racional positivo q e $\tau \neq \iota$

  Fixando um elemento $w \in W$ é interessante verificar que existe uma vizinhança à volta de w na relação R^* cujo valor de m_K aplicado a w e aos elementos dentro da vizinhança é um número é racional e o valor de m_K aplicado a w e aos elementos fora da vizinhança é ι.

- $\Delta = \mathbb{Q}_o^+$ e $\tau \neq \iota$

  Nestas condições, a função m_K nunca toma o valor ι; todos os valores calculados são racionais não negativos.

□

Introduzimos agora a definição de estrutura de interpretação. Basicamente, uma estrutura de interpretação é formada por uma estrutura de Kripke e por um estado. No entanto, a definição da estrutura de Kripke é insuficiente para constituir uma semântica para a lógica. É necessário garantir que o tempo de execução das acções é respeitado e que duas acções não se encontram simultaneamente em execução.

Definição 3.3.5 (Estrutura de Interpretação)
Uma estrutura de interpretação para a assinatura Σ consiste num par $M = \langle K, w\rangle$ onde $K = \langle W, R, V\rangle$ é uma estrutura de Kripke para uma assinatura Σ, $w \in W$ e, se $(att_1, att_2) \in ANM$, verificam-se as seguintes condições para qualquer estado w':

- V_{att_1} e V_{att_2} são aplicações,
- se $V_{att_1}(w') \neq \perp$, então existe $w_1 \in W$ tal que
  - $(w', w_1) \in R^*$,
  - $m_K(w', w_1) \leq_\iota dur(V_{att_1}(w'))$,
  - para qualquer $w_2 \in W$ distinto de w_1,

    se $(w_1, w_2) \in R^*$, então $dur(V_{att_1}(w')) <_\iota m_K(w', w_2)$,
  - $V_{att_2}(w_1) = V_{att_1}(w')$,
  - para qualquer $w_2 \in W$ distinto de w_1,

    se $(w', w_2), (w_2, w_1) \in R^*$, então $V_{att_2}(w_2) = \perp$;
- se $V_{att_2}(w') \neq \perp$, então existe $w_1 \in W$ tal que
  - $(w_1, w') \in R^*$,
  - $m_K(w_1, w') \leq_\iota dur(V_{att_2}(w'))$,
  - para qualquer $w_2 \in W$ distinto de w_1,

    se $(w_2, w_1) \in R^*$, então $dur(V_{att_2}(w')) <_\iota m_K(w_2, w')$,
  - $V_{att_1}(w_1) = V_{att_2}(w')$,
  - para qualquer $w_2 \in W$ distinto de w_1,

    se $(w_1, w_2), (w_2, w') \in R^*$, então $V_{att_1}(w_2) = \perp$.

□

Existem dois tipos de acções: as acções atómicas e as acções com uma duração de execução. As primeiras são caracterizadas por não apresentarem um tempo de execução, não sendo possível separar no tempo o instante em que começou a acção e o instante em que terminou. Suponhamos que o conjunto ANM de uma assinatura contém o par

(att_i, att_f). Ao nível do objecto, estas acções apresentam o seguinte comportamento: se o atributo att_i indica o início da execução da acção, então no mesmo estado, o atributo att_f indica o fim da mesmo acção. Nas acções com uma duração de execução já é possível estabelecer no tempo uma distância entre o início e o fim da sua execução. Este intervalo de execução é constante para a mesma acção, isto é, uma acção não pode apresentar, em duas situações distintas, intervalos de execução distintos. O valor desse intervalo é dado pela aplicação dur. Supondo que uma acção tem um intervalo de execução δ, ao nível do objecto o seu comportamento é o seguinte: se o atributo att_i indica o início da execução da acção, então ao fim de δ unidades de tempo no futuro o atributo att_f indica o seu fim. Analogamente temos que, se att_f indica o fim da execução da acção, então é porque no passado, à distância de δ unidades, o atributo att_i indicou o seu início. O símbolo $\perp$ não é considerado como sendo uma acção porque não pode ser classificado como sendo uma acção atómica ou como sendo uma acção com um intervalo de execução fixo.

Uma outra característica a impor à estrutura de Kripke é a de garantir que duas acções não se encontram simultaneamente em execução. Para isso, o símbolo $\perp$ tem um papel muito importante. Se as acções fossem todas instantâneas, este problema não se levantaria porque os atributos att_i e att_f só poderiam tomar um único valor. Mas na presença de acções com intervalo de execução, o problema torna-se mais complicado. Dizemos que duas acções se encontram em execução simultânea se a ocorrência do fim de execução de uma das acções se verifica no intervalo entre a ocorrência do início e a ocorrência do fim da segunda acção. Para garantir que tal não aconteça, impomos que o atributo att_f tome o valor $\perp$ desde o início da ocorrência da execução de uma acção até se verificar a ocorrência do fim da mesma. O atributo att_i é sujeito à mesma imposição, desde o início da ocorrência da acção até ao seu fim.

3.3.2 Relação de Satisfação

Iremos agora apresentar as condições em que uma estrutura de interpretação satisfaz uma fórmula temporal. Mas antes é necessário introduzir as definições de atribuição a variáveis e de interpretação de termos.

Definição 3.3.6 (Atribuição)
Uma atribuição para as variáveis X consiste numa família $\rho = \{\rho_D : X_D \to D\}_{D \in \mathcal{D}}$ de aplicações.

□

É possível estabelecer uma relação entre duas atribuições.

Prop/Definição 3.3.7 (Relação de Equivalência)
Sejam ρ_1, ρ_2 duas atribuições para as variáveis de X e x uma variável de X_D. Diz-se que ρ_1 é x-equivalente a ρ_2, $\rho_1 \approx_x \rho_2$, sse

- $\rho_{1D'} = \rho_{2D'}$ se $D' \in \mathcal{D}$ e $D' \neq D$,
- $\rho_{1D}(y) = \rho_{2D}(y)$ se $y \in X_D$ e $y \neq x$.

Demonstração:

Facilmente se verifica que a relação $\approx_x$ é reflexiva e simétrica. Sejam ρ_1, ρ_2, ρ_3 três atribuições de variáveis tais que $\rho_1 \approx_x \rho_2, \rho_2 \approx_x \rho_3$ e $y \in X_D$ é uma variável diferente de x. Por definição de x-equivalência temos que $\rho_{1D}(y) = \rho_{2D}(y)$ e $\rho_{2D}(y) = \rho_{3D}(y)$, donde podemos concluir que $\rho_{1D}(y) = \rho_{3D}(y)$, garantindo que $\rho_1 \approx_x \rho_3$.

□

Dada uma atribuição de variáveis é possível definir a interpretação de termos.

Definição 3.3.8 (Interpretação de Termos)
Sejam $M = \langle K, w\rangle$ uma estrutura de interpretação para Σ onde $K = \langle W, R, V\rangle$ e ρ uma atribuição de variáveis. A interpretação de um termo é definida indutivamente do seguinte modo:

- $[\![d]\!]_{\Sigma}^{M,\rho} = d$ se $d \in D$ para $D \in \mathcal{D}$,
- $[\![x]\!]_{\Sigma}^{M,\rho} = \rho_D(x)$ se $x \in X_D$,
- $[\![att]\!]_{\Sigma}^{M,\rho} = V_{att}(w)$,
- $[\![act(t)]\!]_{\Sigma}^{M,\rho} = act([\![t]\!]_{\Sigma}^{M,\rho})$.

□

Definição 3.3.9 (Relação de Satisfação)
Sejam Σ uma assinatura, $M = \langle K, w\rangle$ uma estrutura de interpretação para Σ onde $K = \langle W, R, V\rangle$ e ρ uma atribuição a variáveis. A relação de satisfação $\Vdash_{\Sigma}$ de uma fórmula de F_{Σ} por M e ρ é definida indutivamente como se segue:

- $M, \rho \Vdash_{\Sigma} (t = t')$ sse $[\![t]\!]_{\Sigma}^{M,\rho} = [\![t']\!]_{\Sigma}^{M,\rho}$,
- $M, \rho \Vdash_{\Sigma} (\neg\, \varphi)$ sse $M, \rho \nVdash_{\Sigma} \varphi$,
- $M, \rho \Vdash_{\Sigma} (\varphi_1 \Rightarrow \varphi_2)$ sse $M, \rho \nVdash_{\Sigma} \varphi_1$ ou $M, \rho \Vdash_{\Sigma} \varphi_2$,
- $M, \rho \Vdash_{\Sigma} (\forall x\ \varphi)$ sse $M, \rho' \Vdash_{\Sigma} \varphi$ para qualquer atribuição $\rho' \approx_x \rho$,
- $M, \rho \Vdash_{\Sigma} (\mathsf{X}\, \varphi)$ sse $\langle K, w_1\rangle, \rho \Vdash_{\Sigma} \varphi$ para $w_1 \in W$ tal que $(w, w_1) \in R$,
- $M, \rho \Vdash_{\Sigma} (\varphi_1\ \mathsf{U}_{\delta}\ \varphi_2)$ sse existe $w_1 \in W$ tal que:
  - $(w, w_1) \in R^*$,
  - $m_K(w, w_1) \leq_{\iota} \delta$,

- para o todo $w_2 \in W$ distinto de w_1,

$$\text{se } (w_1, w_2) \in R^* \text{ então } \delta <_\iota m_K(w, w_2),$$

- $\langle K, w_1\rangle, \rho \Vdash_\Sigma \varphi_2$,
- para o todo $w_2 \in W$ distinto de w_1,

$$\text{se } (w, w_2), (w_2, w_1) \in R^* \text{ então } \langle K, w_2\rangle, \rho \Vdash_\Sigma \varphi_1,$$

- $M, \rho \Vdash_\Sigma (\varphi_1 \,\mathsf{U}_{\preceq\delta}\, \varphi_2)$ sse existe $w_1 \in W$ tal que:
  - $(w, w_1) \in R^*$,
  - $m_K(w, w_1) \leq \delta$,
  - $\langle K, w_1\rangle, \rho \Vdash_\Sigma \varphi_2$,
  - para o todo $w_2 \in W$ distinto de w_1,

$$\text{se } (w, w_2), (w_2, w_1) \in R^* \text{ então } \langle K, w_2\rangle, \rho \Vdash_\Sigma \varphi_1,$$

- $M, \rho \Vdash_\Sigma (\mathsf{Y}\, \varphi)$ sse $\langle K, w_1\rangle, \rho \Vdash_\Sigma \varphi$ para $w_1 \in W$ tal que $(w_1, w) \in R$,
- $M, \rho \Vdash_\Sigma (\varphi_1 \,\mathsf{S}_\delta\, \varphi_2)$ sse existe $w_1 \in W$ tal que:
  - $(w_1, w) \in R^*$,
  - $m_K(w_1, w) \leq \delta$,
  - para o todo $w_2 \in W$ distinto de w_1,

$$\text{se } (w_2, w_1) \in R^* \text{ então } \delta <_\iota m_K(w_2, w),$$

  - $\langle K, w_1\rangle, \rho \Vdash_\Sigma \varphi_2$,
  - para o todo $w_2 \in W$ distinto de w_1,

$$\text{se } (w_1, w_2), (w_2, w) \in R^* \text{ então } \langle K, w_2\rangle, \rho \Vdash_\Sigma \varphi_1,$$

- $M, \rho \Vdash_\Sigma (\varphi_1 \,\mathsf{S}_{\preceq\delta}\, \varphi_2)$ sse existe $w_1 \in W$ tal que:
  - $(w_1, w) \in R^*$,
  - $m_K(w_1, w) \leq \delta$,
  - $\langle K, w_1\rangle, \rho \Vdash_\Sigma \varphi_2$,
  - para o todo $w_2 \in W$ distinto de w_1,

$$\text{se } (w_1, w_2), (w_2, w) \in R^* \text{ então } \langle K, w_2\rangle, \rho \Vdash_\Sigma \varphi_1.$$

$\square$

As condições de satisfação para os operadores temporais clássicos têm uma interpretação semelhante na lógica pura ([Fitting 83, Goldblatt 87]).

Uma fórmula envolvendo o operador U_δ é satisfeita se existe um estado, cuja distância ao estado actual seja o mais aproximado possível de δ, que satisfaça a fórmula φ_2 e se todos os estados intermédios, incluindo o estado actual, satisfazem a fórmula φ_1. Para o operador $\mathsf{U}_{\preceq\delta}$ é suficiente que o estado onde se verifica a fórmula φ_2 se encontre a uma distância máxima menor ou igual a δ.

As fórmulas com operadores relacionados com o passado, S_δ e $\mathsf{S}_{\preceq\delta}$, são interpretadas de modo semelhante ao das fórmulas anteriores, mas olhando para o passado.

A relação de satisfação pode ser facilmente estendida a conjuntos de fórmulas.

Notação 3.3.10 (Relação de Satisfação de Conjuntos)
Uma estrutura de interpretação M e uma atribuição ρ satisfazem um conjunto de fórmulas $\Phi \subset F_\Sigma$, o que se escreve $M, \rho \Vdash_\Sigma \Phi$, sse $M, \rho \Vdash_\Sigma \varphi$ para toda a fórmula $\varphi \in \Phi$.

□

Em certas condições, temos a seguinte equivalência:

Proposição 3.3.11 Sejam φ uma fórmula de F_Σ, M uma estrutura de interpretação para a assinatura Σ, ρ uma atribuição a variáveis e x uma variável de X_D. Para qualquer atribuição $\rho' \approx_x \rho$, verifica-se a seguinte equivalência:

$M, \rho \Vdash_\Sigma \varphi$ sse $M, \rho' \Vdash_\Sigma [\varphi]^x_{\rho_D(x)}$.

Demonstração:
A equivalência é facilmente verificada por indução na complexidade da fórmula φ.

□

Em seguida apresentamos alguns resultados clássicos obtidos a partir da noção de satisfação semântica. O mais importante consiste na noção de consequência semântica.

Definição 3.3.12 (Satisfação e Modelo)
Seja Σ uma assinatura de objecto. Uma fórmula φ de F_Σ diz-se satisfeita pela estrutura de interpretação M, o que se escreve $M \Vdash_\Sigma \varphi$, sse $M, \rho \Vdash_\Sigma \varphi$ para toda a atribuição de variáveis ρ. Diz-se também que M é modelo de φ se $M \Vdash_\Sigma \varphi$.

□

Definição 3.3.13 (Conjunto Possível)
Um subconjunto Φ de F_Σ diz-se possível sse existe uma estrutura de interpretação M que possibilite todas as fórmulas do conjunto, isto é, se $M \Vdash_\Sigma \Phi$.

□

Definição 3.3.14 (Conjunto Contraditório)
Um subconjunto Φ de F_Σ diz-se contraditório sse, para qualquer estrutura de interpretação M e atribuição a variáveis ρ, existe uma fórmula φ de Φ tal que $M, \rho \nVdash_\Sigma \varphi$.

□

Definição 3.3.15 (Consequência Semântica)
Sejam Σ uma assinatura de objecto, Φ um subconjunto finito de F_Σ e $\varphi \in F_\Sigma$. Diz-se que φ é consequência semântica de Φ, o que se escreve $\Phi \vDash_\Sigma \varphi$, sse para toda a estrutura de interpretação M, se $M \Vdash_\Sigma \Phi$ então $M \Vdash_\Sigma \varphi$.

□

Proposição 3.3.16 (Consequência Semântica vs. Conjunto Contraditório)
Sejam Σ uma assinatura de objecto, Φ um subconjunto finito de F_Σ e $\varphi \in F_\Sigma$. A fórmula φ é consequência semântica do conjunto Φ sse o conjunto $\Phi \cup \{(\neg\,\varphi)\}$ é contraditório.

Demonstração:
Verifiquemos os sentidos da equivalência:

$\rightarrow$ Suponhamos que $\Phi \vDash_\Sigma \varphi$ mas que o conjunto $\Phi \cup \{(\neg\,\varphi)\}$ é possível, isto é, existe uma estrutura de interpretação M tal que $M \Vdash_\Sigma \Phi \cup \{(\neg\,\varphi)\}$. Como $M \Vdash_\Sigma \Phi$, por definição de satisfação semântica, temos que $M \Vdash_\Sigma \varphi$. Mas $M \Vdash_\Sigma (\neg\,\varphi)$, o que é impossível.

$\leftarrow$ Suponhamos que o conjunto $\Phi \cup \{(\neg\,\varphi)\}$ é contraditório mas que é falso que $\Phi \vDash_\Sigma \varphi$. Da segunda afirmação podemos obter uma estrutura de interpretação M tal que $M \Vdash_\Sigma \Phi$ e $M \nVdash_\Sigma \varphi$. Por definição de satisfação semântica, existe uma atribuição a variáveis ρ onde $M, \rho \Vdash_\Sigma (\neg\,\varphi)$. Então existe uma estrutura de interpretação e uma atribuição a variáveis que satisfaz o conjunto $\Psi \cup \{(\neg\,\varphi)\}$, o que é impossível visto que esse conjunto é contraditório.

□

Temos então as seguintes noções para especificações de objectos:

Definição 3.3.17 (Especificação de Objecto Coerente)
Seja $Ob = \langle \Sigma, Ax \rangle$ uma especificação de objecto. Diz-se que Ob é coerente sse o conjunto Ax é possível.

□

Definição 3.3.18 (Consequência Semântica de uma Especificação de Objecto)

Sejam $Ob = \langle \Sigma, Ax \rangle$ uma especificação de objecto coerente e φ uma fórmula de F_Σ. Diz-se que φ é consequência semântica de Ob, escrevendo-se $Ob \vDash_\Sigma \varphi$, sse $Ax \vDash_\Sigma \varphi$.

□

Exemplo 3.3.19 Considerando as especificações de objectos apresentadas no exemplo 3.2.11, em particular a especificação do objecto `dist_bin`, podemos verificar que `dist_bin` é coerente. De facto, existe um modelo para os axiomas de `dist_bin`. Seja $M = \langle K, w_0 \rangle$ uma estrutura de interpretação onde $K = \langle W, R, V \rangle$ apresenta a seguinte definição:

- $W = \{w_n \ : n \in I\!N_o\}$,
- $R = \{(w_n, w_{n+1}) \ : n \in I\!N_o\}$,
- $V_{occ_i}(w) = V_{occ_f}(w) = \perp$ para qualquer $w \in W$,

verifica-se que $M \Vdash_\Sigma Ax_{\texttt{dist\_bin}}$.

Supondo que um pacote seria danificado se a patilha alterasse a sua posição no intervalo desde a entrada do pacote até quatro unidades de tempo depois, iremos verificar que:

$$\texttt{dist\_bin} \vDash_{\Sigma_{dist\_bin}} (((occ_i = \texttt{entr\_pact}(x)) \wedge (\neg(\mathsf{F}_{\preceq 4}((occ_i = \texttt{saída\_pact}_1(y)) \vee$$

$$(occ_i = \texttt{saída\_pact}_2(y)))))) \Rightarrow (\neg(\mathsf{F}_{\preceq 4}(occ_i = \texttt{md\_pat}))))$$

Por absurdo, seja $M = \langle K, w_0 \rangle$ uma estrutura de interpretação que satisfaz os axiomas da especificação mas não satisfaz a implicação. Temos então que existe uma atribuição ρ tal que:

$$M, \rho \Vdash_{\Sigma_{dist\_bin}} (occ_i = \texttt{entr\_pact}(x)),$$

$$M, \rho \Vdash_{\Sigma_{dist\_bin}} (\neg(\mathsf{F}_{\preceq 4}((occ_i = \texttt{saída\_pact}_1(y)) \vee (occ_i = \texttt{saída\_pact}_2(y))))),$$

$$M, \rho \Vdash_{\Sigma_{dist\_bin}} (\mathsf{F}_{\preceq 4}(occ_i = \texttt{md\_pat})).$$

Da terceira relação de satisfação, podemos obter um dos seguintes casos:

- $V_{occ_i}(w_0) = \texttt{md\_pat}$,
- $V_{occ_i}(w_1) = \texttt{md\_pat}$ onde $(w_0, w_1) \in R$,
- $V_{occ_i}(w_2) = \texttt{md\_pat}$ onde $(w_1, w_2) \in R$.

O primeiro caso entra em contradição com a primeira relação de satisfação, uma vez que $V_{occ_i}(w_0) = \texttt{entr\_pact}(\rho(x))$.

Para verificar o segundo caso, teríamos que, segundo o axioma $db6$, $V_{occ_i}(w_0) = \texttt{saída\_pact}_1(\rho(x))$ ou $V_{occ_i}(w_0) = \texttt{saída\_pact}_2(\rho(x))$, o que entra em contradição com a primeira relação de satisfação.

Finalmente, e recorrendo ao axioma anterior, o terceiro caso implicaria que $V_{occ_i}(w_1) = \texttt{saída\_pact}_1(\rho(x))$ ou $V_{occ_i}(w_1) = \texttt{saída\_pact}_2(\rho(x))$, o que entra em contradição com a segunda relação de satisfação.

Podemos concluir que a consequência semântica é verdadeira.

□

Capítulo 4

Sistema Dedutivo: Decidibilidade

4.1 Introdução

Neste capítulo iremos definir um sistema dedutivo para a lógica apresentada no capítulo anterior. Os sistema dedutivo será baseado em tableaux declarativos. Para ser possível raciocinar com tableaux declarativos, é necessário enriquecer as fórmulas temporais com prefixos. Esses prefixos irão representar os estados. Uma fórmula é prefixada por um dado prefixo se pretendermos que essa fórmula seja satisfeita no estado representado por esse prefixo. A técnica de prefixar fórmulas foi introduzida por Fitting em [Fitting 83] e utilizada em [Resende 91, Gouveia 92, Gouveia *et al* 93]. Faremos também a introdução do predicado binário α sobre os prefixos para representar a relação de visibilidade directa sobre os estados representados pelos prefixos. No entanto, estas técnicas não são suficientes para desenvolver um sistema de tableaux que seja correcto e completo. É necessário garantir que não existem variáveis livres nas fórmulas prefixadas. Em [Gallier 87] encontra-se a definição de fecho universal de uma fórmula. Este fecho será utilizado para tornar mudas todas as variáveis livres de uma fórmula.

Notação 4.1.1 (Fecho Universal)
Seja φ uma fórmula de F_Σ. O seu fecho universal $\varphi\prime$ obtem-se quantificando universalmente todas as variáveis que ocorrem livres em φ.

□

Um resultado importante apresentado em [Gallier 87] é o seguinte:

Proposição 4.1.2 (Equivalência no Fecho Universal)
Sejam φ uma fórmula de F_Σ e M uma estrutura de interpretação para a assinatura Σ. Verifica-se a seguinte equivalência:

$$M \Vdash_\Sigma \varphi \text{ sse } M \Vdash_\Sigma \varphi\prime.$$

Demonstração:
Consideremos o caso particular em que $vl_\Sigma(\varphi) = \{x\}$. Então, $M \Vdash_\Sigma (\forall x\ \varphi)$ sse $M, \rho \Vdash_\Sigma (\forall x\ \varphi)$ para todo a atribuição ρ, e ainda, por definição 3.3.9 de relação de satisfação, sse $M, \rho' \Vdash_\Sigma \varphi$ para todo $\rho' \approx_x \rho$. Uma vez que estamos a considerar uma atribuição ρ qualquer, podemos deduzir que $M, \rho \Vdash_\Sigma \varphi$ para toda a atribuição ρ, o que é equivalente a dizer que $M \Vdash_\Sigma \varphi$.

□

A correcção e a completude do sistema de tableaux serão provadas mais à frente. No final do capítulo será apresentado um algoritmo de decisão para o sistema.

4.2 Fórmulas Prefixadas

Ao prefixar as fórmulas temporais, o objectivo é introduzir directamente numa fórmula um representante do estado onde essa fórmula é satisfeita. Para tal, iremos enriquecer a definição de assinatura de objecto com um conjunto numerável de prefixos. Cada prefixo representa um único estado. Uma relação entre os prefixos é necessária para importar a relação de visibilidade.

4.2.1 Sintaxe

Definição 4.2.1 (Assinatura de Prefixos)
Uma assinatura de prefixos é um triplo $\Gamma = \langle \Sigma, \Omega, \alpha \rangle$ onde Σ é uma assinatura de objecto, Ω é um conjunto numerável de elementos intitulados prefixos e α é um predicado sobre $\Omega \otimes \Omega$.

□

Definição 4.2.2 (Fórmulas Prefixadas)
Seja $\Gamma = \langle \Sigma, \Omega, \alpha \rangle$ uma assinatura de prefixos. O conjunto das fórmulas prefixadas é definido como se segue:

$$F_\Gamma = \{(\omega_1 \alpha \omega_2) :\ \omega_1, \omega_2 \in \Omega\} \cup \{(\omega : \varphi\prime) :\ \omega \in \Omega \text{ e } \varphi \in F_\Sigma\}.$$

□

Existem dois tipos de fórmulas temporais prefixadas: umas representam a relação de visibilidade directa, ao passo que outras representam a satisfação de uma fórmula temporal num estado representado por um prefixo. Para estas últimas, iremos recorrer ao seu fecho universal para não ser necessário ter em mente uma atribuição a variáveis específica.

Para facilitar futuros raciocínios, definimos a notação seguinte:

Notação 4.2.3 Sejam $\Psi \subseteq F_\Gamma$ e $\omega, \omega' \in \Omega$.

- Diz-se que $(\omega\alpha^n\omega') \in \Psi$ para $n \in I\!N_0$ sse existem prefixos $\omega_0, \ldots, \omega_n \in \Omega$ tais que:
  - $\omega_0 = \omega, \omega_n = \omega'$,

    e
  - $\{(\omega_0\alpha\omega_1), \ldots, (\omega_{n-1}\alpha\omega_n)\} \subseteq \Psi$.
- Diz-se que $(\omega\alpha^*\omega') \in \Psi$ sse existe $n \in I\!N_o$ tal que $(\omega\alpha^n\omega') \in \Psi$.

$\square$

Ao fixarmos um prefixo ω, facilmente se passa de uma fórmula não prefixada para uma fórmula prefixada ou mesmo de um conjunto de fórmulas não prefixadas para um conjunto de fórmulas prefixadas.

Notação 4.2.4 (Prefixação de uma Fórmula ou de um Conjunto de Fórmulas)

Sejam ω um prefixo de Γ, φ uma fórmula de F_Σ e Φ um conjunto de fórmulas de F_Σ. A prefixação de φ por ω é definida do seguinte modo:

$$\varphi_\omega = (\omega : \varphi\prime).$$

A prefixação de um conjunto é determinada por:

$$\Phi_\omega = \{\phi_\omega \ : \ \phi \in \Phi\}.$$

$\square$

Na definição das regras de inferência em 4.3.2, é necessário determinar os prefixos que se encontram numa fórmula prefixada.

Definição 4.2.5 (Prefixos de uma Fórmula)
A aplicação $pref : F_\Gamma \to 2^\Omega$ tal que:

- $pref(\omega_1\alpha\omega_2) = \{\omega_1, \omega_2\}$,
- $pref(\omega : \varphi) = \{\omega\}$,

determina os prefixos existentes numa fórmula de F_Γ.

□

É possível estender a aplicação de prefixos a conjuntos de fórmulas. Seja $\Psi \subseteq F_\Gamma$. Temos que

$$pref(\Psi) = \bigcup_{\psi \in \Psi} pref(\psi).$$

4.2.2 Semântica

Uma assinatura de prefixos é um enriquecimento de uma assinatura de objectos. De igual modo, a semântica para uma assinatura de prefixos será um enriquecimento de uma estrutura de Kripke para a assinatura de objecto subjacente. No entanto, é necessário que a estrutura de Kripke verifique as propriedades relacionadas com o tempo de execução das acções impostas na definição 3.3.5 de estrutura de interpretação.

Definição 4.2.6 (Interpretação de Assinatura de Prefixos)
Seja $\Gamma = \langle \Sigma, \Omega, \alpha \rangle$ uma assinatura de prefixos. Uma interpretação I_p de Γ é um par $\langle K, S \rangle$ onde:

- $K = \langle W, R, V \rangle$ é uma estrutura de Kripke para a assinatura Σ que respeita o tempo de execução das acções, isto é, $\langle K, w \rangle$ é uma estrutura de interpretação de Σ para qualquer estado $w \in W$,
- $S : \Omega \to W$ é uma aplicação que associa um estado a um prefixo.

□

A satisfação de uma fórmula prefixada fica facilitada com a semântica apresentada. É apenas necessário fazer a ponte de uma fórmula prefixada para uma fórmula não prefixada e recorrer à relação de satisfação apresentada em 3.3.9.

Definição 4.2.7 (Satisfação de uma Fórmula Prefixada)
Sejam Σ uma assinatura de objecto, Γ uma assinatura de prefixos e $I_p = \langle K, S \rangle$ uma interpretação de Γ onde $K = \langle W, R, V \rangle$. Diz-se que uma fórmula ψ de F_Γ é satisfeita por I_p, escrevendo-se $I_p \Vdash_\Gamma \psi$, sse forem verificadas as seguintes condições:

- se $\psi = (\omega_1 \alpha \omega_2)$ então $(S(\omega_1), S(\omega_2)) \in R$,
- se $\psi = (\omega : \varphi)$ então $\langle K, S(\omega) \rangle \Vdash_\Sigma \varphi$.

□

É possível estender a noção de satisfação a um conjunto de fórmulas.

Definição 4.2.8 (Satisfação de um Conjunto de Fórmulas)
Sejam Σ uma assinatura de objecto e Γ uma assinatura de prefixos. Diz-se que um subconjunto Ψ de F_Γ é satisfeito pela interpretação I_p, escrevendo-se $I_p \Vdash_\Gamma \Psi$, sse $I_p \Vdash_\Gamma \psi$ para todo o $\psi \in \Psi$.

□

Estabelecemos em seguida uma equivalência entre a satisfação de fórmulas prefixadas e a satisfação de fórmulas não prefixadas.

Proposição 4.2.9 (Equivalência Entre Satisfações)
Sejam $\Gamma = \langle \Sigma, \Omega, \alpha \rangle$ uma assinatura de prefixos, $I_p = \langle K, S \rangle$ uma interpretação de Γ, φ uma fórmula de F_Σ e ω um prefixo de Ω. Temos que:

$\langle K, S(\omega) \rangle \Vdash_\Sigma \varphi$ sse $I_p \Vdash_\Gamma \varphi_\omega$.

Demonstração:

$\rightarrow$ Suponhamos que $\langle K, S(\omega) \rangle \Vdash_\Sigma \varphi$. Pela proposição 4.1.2, temos que $\langle K, S(\omega) \rangle \Vdash_\Sigma \varphi\prime$, de onde se conclui que $I_p \Vdash_\Sigma (\omega : \varphi\prime)$.

$\leftarrow$ Por definição de satisfação semântica, de $I_p \Vdash_\Sigma \varphi_\omega$ podemos tirar que $\langle K, S(\omega) \rangle \Vdash_\Sigma \varphi\prime$. Como $\varphi\prime$ é o fecho universal de φ, podemos concluir que $\langle K, S(\omega) \rangle \Vdash_\Sigma \varphi$.

□

Apresentamos agora algumas definições e proposições relacionadas com a noção de satisfação.

Definição 4.2.10 (Conjunto Possível)
Seja Ψ um conjunto de fórmulas de F_Γ. Diz-se que Ψ é possível sse existe uma interpretação I_p para Γ tal que $I_p \Vdash_\Gamma \Psi$.

□

Definição 4.2.11 (Conjunto Contraditório)
Seja Ψ um conjunto de fórmulas de F_Γ. Diz-se que Ψ é contraditório sse, para toda a interpretação I_p de Γ, existe uma fórmula ψ de Ψ tal que $I_p \nVdash_\Gamma \psi$.

□

Definição 4.2.12 (Consequência Semântica)
Sejam Ψ um conjunto de F_Γ e ψ uma fórmula de F_Γ. Diz-se que ψ é consequência semântica de Ψ, escrevendo-se $\Psi \vDash_\Gamma \psi$, sse, para todas as interpretação I_p de Γ, se $I_p \Vdash_\Gamma \Psi$ então $I_p \Vdash_\Gamma \psi$.

□

Proposição 4.2.13 (Consequência Semântica vs Conjunto Contraditório)
Sejam Ψ um conjunto de fórmulas de F_Γ e $\psi = (\omega : \varphi)$ uma fórmula de F_Γ. A fórmula ψ é consequência semântica do conjunto Ψ sse o conjunto $\Psi \cup \{(\omega : (\neg \varphi))\}$ é um conjunto contraditório.

Demonstração:
A demonstração desta proposição é semelhante à demonstração da proposição 3.3.16.

□

Entre as duas definições de consequência semântica, 3.3.15 e 4.2.12, é possível estabelecer a seguinte equivalência:

Proposição 4.2.14 (Equivalência entre Consequências Semânticas)
Sejam $\Gamma = \langle \Sigma, \Omega, \alpha \rangle$ uma assinatura de prefixos, Φ um subconjunto de F_Σ, φ uma fórmula de F_Σ e $\omega \in \Omega$. Verifica-se a seguinte equivalência:

$\Phi \vDash_\Sigma \varphi$ sse $\Phi_\omega \vDash_\Gamma \varphi_\omega$.

Demonstração:
A demonstração é trivial recorrendo à proposição 4.2.9.

□

Para finalizar esta secção, seguindo a linha de definições do capítulo anterior, indicamos quando é que uma fórmula prefixada é consequência de uma especificação de objecto.

Definição 4.2.15 (Consequência de uma Especificação de Objecto)
Sejam $Ob = \langle \Sigma, Ax \rangle$ uma especificação de objecto e $(\omega : \varphi)$ uma fórmula de F_Γ. Diz-se que $(\omega : \varphi)$ é consequência de Ob, escrevendo-se $Ob \vDash_\Gamma (\omega : \varphi)$, sse $Ax_\omega \vDash_\Gamma (\omega : \varphi)$.

□

Com estes resultados sobre fórmulas prefixadas, verificamos que, com algumas restrições, é possível estabelecer uma ligação entre a satisfação de fórmulas prefixadas e a satisfação de fórmulas não prefixadas. Existe mesmo uma equivalência entre as definições de consequência semântica. Ao longo deste capítulo, iremos considerar apenas as fórmulas prefixadas, mas teremos sempre em consideração que existe a possibilidade de estabelecer uma satisfação ou uma consequência semântica tal como foram definidas no capítulo anterior.

Sendo uma assinatura de prefixos o enriquecimento de uma assinatura de objecto, ao apresentar uma assinatura de objecto sem especificar a assinatura de prefixo, subentende-se que o conjunto de prefixos Ω seja $\{\omega_n : n \in I\!N_o\}$.

4.3 Sistema de Tableaux

Um sistema de tableaux é num par $\langle Ab, RI\rangle$ onde Ab é num conjunto de absurdos e RI um conjunto de regras de inferência. Nesta secção será definido o sistema de tableaux para uma assinatura de objecto $\Sigma = \langle \Delta, \tau, GACT, GATT\rangle$.

Definição 4.3.1 (Absurdo)
Um subconjunto Ψ de F_Γ diz-se um absurdo sse verifica uma das seguintes condições:

- $\{(\omega : \varphi), (\omega : (\neg\,\varphi))\} \subseteq \Psi$, em que $\omega \in \Omega$ e $\varphi \in F_\Sigma$,
- $(\omega : (\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2)) \in \Psi$ e para qualquer natural n existe um prefixo $w' \in \Omega$ tal que $\{(\omega\alpha^n\omega'), (\omega' : (\neg\,\varphi_2))\} \subseteq \Psi$, em que $\varphi_1, \varphi_2 \in F_\Sigma$ e $\omega \in \Omega$,
- $\omega \in pref(\Psi)$ e para qualquer natural n existe um prefixo ω' em que $(\omega'\alpha^n\omega) \in \Psi$.

□

A primeira condição é a usual na definição de absurdos num sistema de tableaux. As restantes já apresentam algumas inovações. Semanticamente, quando uma fórmula do tipo $(\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2)$ é satisfeita num estado w, então existe um estado w' onde se verifica a fórmula φ_2 e $(w, w') \in R^*$. A nível sintáctico, o absurdo relacionado com esta fórmula é o seguinte: a fórmula $(\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2)$ verifica-se no estado representado pelo prefixo ω, mas para todos os estados representados pelos prefixos ω' no futuro do estado anterior, a fórmula φ_2 não é verificada. Esta propriedade encontra-se expressa através da segunda condição da definição de absurdo. A terceira condição de absurdo caracteriza o caso em que existe um passado infinito sendo, portanto, impossível definir um estado inicial.

Designamos o conjunto dos absurdos de Σ por Ab_Σ.

Definição 4.3.2 (Regras de Inferência)
Sejam $\Psi \subseteq F_\Gamma$ um conjunto de fórmulas, $\varphi, \varphi_1, \varphi_2 \in F_\Sigma$ fórmulas, $\omega, \omega_1, \omega_2 \in \Omega$ prefixos e

$t, t_1, t_3 \in T_{\Sigma,D}$ termos para algum domínio D. As regras de inferência para a assinatura Σ, RI_Σ, são:

1. $= i$

   $\Psi, (\omega : (t = t))$

   $\therefore\ \Psi$

2. $= ii$

   $\Psi, (\omega : (t_1 = t_2)), (\omega : (t_2 = t_1))$

   $\therefore\ \Psi, (\omega : (t_1 = t_2))$

3. $= iii$

   $\Psi, (\omega : (t_1 = t_2)), (\omega : (t_2 = t_3)), (\omega : (t_1 = t_3))$

   $\therefore\ \Psi, (\omega : (t_1 = t_2)), (\omega : (t_2 = t_3))$

4. $= iv$

   $\Psi, (\omega : (t = t')), (\omega : (\neg(t = t_1))), \ldots, (\omega : (\neg(t = t_n)))$

   $\therefore\ \Psi, (\omega : (t = t'))$

   se $T_{\Sigma,D} = \{t, t', t_1, \ldots, t_n\}$ e $n \in I\!N$,

5. $\Rightarrow$

   $\Psi, (\omega : (\neg\, \varphi_1))$;

   $\Psi, (\omega : \varphi_2)$

   $\therefore\ \Psi, (\omega : (\varphi_1 \Rightarrow \varphi_2))$

6. $\neg \Rightarrow$

   $\Psi, (\omega : \varphi_1), (\omega : (\neg\, \varphi_2))$

   $\therefore\ \Psi, (\omega : (\neg(\varphi_1 \Rightarrow \varphi_2)))$

7. $\neg\, \neg$

   $\Psi, (\omega : \varphi)$

   $\therefore\ \Psi, (\omega : (\neg(\neg\, \varphi)))$

8. $\forall$

   $\Psi, (\omega : [\varphi]^x_{d_1}), \ldots, (\omega : [\varphi]^x_{d_n})$

   $\therefore\ \Psi, (\omega : (\forall x\ \varphi))$

   se $x \in X_D$ e $D = \{d_1, \ldots, d_n\}$,

9. $\neg\,\forall$

$$\Psi, (\omega : (\neg[\varphi]^x_{d_1}))\ ;$$
$$\vdots$$
$$\Psi, (\omega : (\neg[\varphi]^x_{d_n}))$$
$$\therefore\ \Psi, (\omega : (\neg(\forall x\ \varphi)))$$

se $x \in X_D$ e $D = \{d_1, \ldots, d_n\}$,

10. $\mathsf{X}\,i$

$$\Psi, (\omega_1 \alpha \omega_2), (\omega_2 : \varphi)$$
$$\therefore\ \Psi, (\omega_1 : (\mathsf{X}\,\varphi))$$

se

- $\omega_2 \notin pref(\Psi) \cup \{\omega_1\}$,
- $(\omega_1 \alpha \omega) \notin \Psi$ para qualquer $\omega \in \Omega$

11. $\mathsf{X}\,ii$

$$\Psi, (\omega_1 \alpha \omega_2), (\omega_2 : \varphi)$$
$$\therefore\ \Psi, (\omega_1 \alpha \omega_2), (\omega_1 : (\mathsf{X}\,\varphi))$$

12. $\neg\,\mathsf{X}$

$$\Psi, (\omega_1 \alpha \omega_2), (\omega_2 : (\neg\,\varphi))$$
$$\therefore\ \Psi, (\omega_1 \alpha \omega_2), (\omega : (\neg(\mathsf{X}\,\varphi)))$$

13. $\mathsf{U}_\delta i$

$$\Psi, (\omega : \varphi_2)$$
$$\therefore\ \Psi, (\omega : (\varphi_1\ \mathsf{U}_{\delta_1}\ \varphi_2))$$

se $\delta_1 <_\iota \tau$,

14. $\mathsf{U}_\delta ii$

$$\Psi, (\omega : \varphi_1), (\omega : (\mathsf{X}(\varphi_1\ \mathsf{U}_{\delta_1 -_\iota \tau}\ \varphi_2)))$$
$$\therefore\ \Psi, (\omega : (\varphi_1\ \mathsf{U}_{\delta_1}\ \varphi_2))$$

se $\tau \leq_\iota \delta_1$,

15. $\neg\,\mathsf{U}_\delta i$

$$\Psi, (\omega : (\neg\,\varphi_2))$$
$$\therefore\ \Psi, (\omega : (\neg(\varphi_1\ \mathsf{U}_{\delta_1}\ \varphi_2)))$$

se $\delta_1 <_\iota \tau$,

16. $\neg\,\mathsf{U}_\delta ii$

    $\Psi, (\omega : (\neg\,\varphi_1))\ ;$

    $\Psi, (\omega : (\neg(\mathsf{X}(\varphi_1\ \mathsf{U}_{\delta_1 -_\iota \tau}\ \varphi_2))))$

    $\therefore\ \Psi, (\omega : (\neg(\varphi_1\ \mathsf{U}_{\delta_1}\ \varphi_2)))$

    se $\tau \leq_\iota \delta_1$,

17. $\mathsf{U}_{\preceq\delta} i$

    $\Psi, (\omega : \varphi_2)$

    $\therefore\ \Psi, (\omega : (\varphi_1\ \mathsf{U}_{\preceq\delta_1}\ \varphi_2))$

    se $\delta_1 <_\iota \tau$,

18. $\mathsf{U}_{\preceq\delta} ii$

    $\Psi, (\omega : \varphi_2)\ ;$

    $\Psi, (\omega : (\neg\,\varphi_2)), (\omega : \varphi_1), (\omega : (\mathsf{X}(\varphi_1\ \mathsf{U}_{\preceq\delta_1 -_\iota \tau}\ \varphi_2)))$

    $\therefore\ \Psi, (\omega : (\varphi_1\ \mathsf{U}_{\preceq\delta_1}\ \varphi_2))$

    se $\tau \leq_\iota \delta_1$,

19. $\neg\,\mathsf{U}_{\preceq\delta} i$

    $\Psi, (\omega : (\neg\,\varphi_2))$

    $\therefore\ \Psi, (\omega : (\neg(\varphi_1\ \mathsf{U}_{\preceq\delta_1}\ \varphi_2)))$

    se $\delta_1 <_\iota \tau$,

20. $\neg\,\mathsf{U}_{\preceq\delta} ii$

    $\Psi, (\omega : (\neg\,\varphi_1)), (\omega : (\neg\,\varphi_2))\ ;$

    $\Psi, (\omega : (\neg\,\varphi_2)), (\omega : (\neg(\mathsf{X}(\varphi_1\ \mathsf{U}_{\preceq\delta_1 -_\iota \tau}\ \varphi_2))))$

    $\therefore\ \Psi, (\omega : (\neg(\varphi_1\ \mathsf{U}_{\preceq\delta_1}\ \varphi_2)))$

    se $\delta_1 \leq_\iota \tau$,

21. $\mathsf{Y}\, i$

    $\Psi, (\omega_1 \alpha \omega_2), (\omega_1 : \varphi)$

    $\therefore\ \Psi, (\omega_2 : (\mathsf{Y}\,\varphi))$

    se

    - $\omega_1 \notin pref(\Psi) \cup \{\omega_2\}$,
    - $(\omega\alpha\omega_2) \notin \Psi$ para qualquer $\omega \in \Omega$.

22. $\mathsf{Y}\, ii$

$$\Psi, (\omega_1 \alpha \omega_2), (\omega_1 : \varphi)$$
$$\therefore\ \Psi, (\omega_1 \alpha \omega_2), (\omega_2 : (\mathsf{Y}\, \varphi));$$

23. $\neg\, \mathsf{Y}$

$$\Psi, (\omega_1 \alpha \omega_2), (\omega_1 : (\neg\, \varphi))$$
$$\therefore\ \Psi, (\omega_1 \alpha \omega_2), (\omega_2 : (\neg (\mathsf{Y}\, \varphi)));$$

24. $\mathsf{S}_\delta i$

$$\Psi, (\omega : \varphi_2)$$
$$\therefore\ \Psi, (\omega : (\varphi_1\, \mathsf{S}_{\delta_1}\, \varphi_2))$$

se $\delta_1 <_\iota \tau$,

25. $\mathsf{S}_\delta ii$

$$\Psi, (\omega : \varphi_1), (\omega : (\mathsf{Y}(\varphi_1\, \mathsf{S}_{\delta_1 -_\iota \tau}\, \varphi_2)))$$
$$\therefore\ \Psi, (\omega \alpha \omega_2), (\omega : (\varphi_1\, \mathsf{S}_\delta\, \varphi_2))$$

se $\tau \leq_\iota \delta_1$,

26. $\neg\, \mathsf{S}_\delta i$

$$\Psi, (\omega : (\neg\, \varphi_2))$$
$$\therefore\ \Psi, (\omega : (\neg (\varphi_1\, \mathsf{S}_{\delta_1}\, \varphi_2)))$$

se $\delta_1 <_\iota \tau$,

27. $\neg\, \mathsf{S}_\delta ii$

$$\Psi, (\omega : (\neg\, \varphi_1))\ ;$$
$$\Psi, (\omega : (\neg (\mathsf{Y}(\varphi_1\, \mathsf{S}_{\delta_1 -_\iota \tau}\, \varphi_2))))$$
$$\therefore\ \Psi, (\omega : (\neg (\varphi_1\, \mathsf{S}_{\delta_1}\, \varphi_2)))$$

se $\tau \leq_\iota \delta_1$,

28. $\mathsf{S}_{\preceq \delta} i$

$$\Psi, (\omega : \varphi_2)$$
$$\therefore\ \Psi, (\omega : (\varphi_1\, \mathsf{S}_{\preceq \delta_1}\, \varphi_2))$$

se $\delta_1 <_\iota \tau$,

29. $\mathsf{S}_{\preceq \delta} ii$

$$\Psi, (\omega : \varphi_2)\ ;$$
$$\Psi, (\omega : (\neg\, \varphi_2)), (\omega : \varphi_1), (\omega : (\varphi_1\, \mathsf{S}_{\preceq \delta_1 -_\iota \tau}\, \varphi_2))$$
$$\therefore\ \Psi, (\omega : (\varphi_1\, \mathsf{S}_{\preceq \delta_1}\, \varphi_2))$$

se $\tau \leq_\iota \delta_1$,

30. $\neg\, \mathsf{S}_{\preceq\delta} i$

$$\Psi, (\omega : (\neg\, \varphi_2))$$

$$\therefore\ \Psi, (\omega : (\neg(\varphi_1\, \mathsf{S}_{\preceq\delta_1}\, \varphi_2)))$$

se $\delta_1 <_\iota \tau$,

31. $\neg\, \mathsf{S}_{\preceq\delta} ii$

$$\Psi, (\omega : (\neg\, \varphi_1)), (\omega : (\neg\, \varphi_2))\ ;$$

$$\Psi, (\omega : (\neg\, \varphi_2)), (\omega : (\neg(\mathsf{Y}(\varphi_1\, \mathsf{S}_{\preceq\delta_1 -_\iota \tau}\, \varphi_2))))$$

$$\therefore\ \Psi, (\omega : (\neg(\varphi_1\, \mathsf{S}_{\preceq\delta_1}\, \varphi_2)))$$

se $\delta_1 \leq_\iota \tau$,

□

As regras de inferência apresentadas são semelhantes às regras de inferência definidas em [Manna e Pnueli 91], mas adaptadas para os novo operadores temporais. O objectivo das regras é decompor uma fórmula recorrendo ao menor número possível de prefixos. Por exemplo, a regra 14 é interpretada do seguinte modo: verifica-se $(\varphi_1\, \mathsf{U}_{\delta_1}\, \varphi_2)$ no estado representado por ω se φ_1 for possível no mesmo estado e se $(\varphi_1\, \mathsf{U}_{\delta_1 -_\iota \tau}\, \varphi_2)$ for possível no estado seguinte. Como existe a possibilidade de não se conhecer o representante do próximo estado, a última fórmula é substituída por $(\mathsf{X}(\varphi_1\, \mathsf{U}_{\delta_1 -_\iota \tau}\, \varphi_2))$ e prefixada por ω. Com esta decomposição, a necessidade de juntar ou não um novo prefixo ao sistema é da responsabilidade exclusiva das regras 10 e 11 para o operador X. Para o passado, o raciocínio é semelhante, mas com os respectivos operadores temporais.

Analisemos mais algumas regras. Para a negação do operador X existe uma única regra, a 12. Neste caso, a fórmula $(\neg(\mathsf{X}\, \varphi))$ é satisfeita no estado representado pelo prefixo ω sse a fórmula φ não é satisfeita no próximo estado. Isto é, ou não existe nenhum estado que sucede ao actual, ou nesse estado é satisfeita a fórmula $(\neg\, \varphi)$. Por isso só faz sentido decompor a fórmula $(\neg(\mathsf{X}\, \varphi))$ se houver a garantia da existência de tal estado.

A regra 13 é um reflexo do tempo discreto escolhido. Se o período do relógio é superior ao valor de tempo associado à fórmula $(\varphi_1\, \mathsf{U}_{\delta_1}\, \varphi_2)$, então para verificar essa fórmula é necessário que φ_2 seja satisfeita no estado actual, uma vez que o estado seguinte se encontra a uma distância superior a δ_1 no tempo.

Com a definição das regras de inferência, fica completa a definição do sistema de tableaux para uma assinatura de objecto Σ.

Definição 4.3.3 (Sistema de Tableaux)
O sistema de tableaux para a assinatura de prefixos Γ é o par $\langle Ab_\Gamma, RI_\Gamma \rangle$.

□

Para as abreviaturas, definimos as seguintes regras de inferência:

32. U

$$\Psi, (\omega : \varphi_2)\ ;$$
$$\Psi, (\neg(\omega : \varphi_2)), (\omega : \varphi_1), (\omega : (\mathsf{X}(\varphi_1 \mathbin{\mathsf{U}} \varphi_2)))$$
$$\therefore\ \Psi, (\omega : (\varphi_1 \mathbin{\mathsf{U}} \varphi_2))$$

33. $\neg\,\mathsf{U}$

$$\Psi, (\omega : (\neg\,\varphi_1)), (\omega : (\neg\,\varphi_2))\ ;$$
$$\Psi, (\omega : (\neg\,\varphi_2)), (\omega : (\neg(\mathsf{X}(\varphi_1 \mathbin{\mathsf{U}} \varphi_2))))$$
$$\therefore \Psi, (\omega : (\neg(\varphi_1 \mathbin{\mathsf{U}} \varphi_2)))$$

34. F

$$\Psi, (\omega : \varphi)\ ;$$
$$\Psi, (\omega : (\neg\,\varphi)), (\omega : (\mathsf{X}(\mathsf{F}\,\varphi)))$$
$$\therefore\ \Psi, (\omega : (\mathsf{F}\,\varphi)).$$

35. $\neg\,\mathsf{F}$

$$\Psi, (\omega : (\neg\,\varphi)), (\omega : (\neg(\mathsf{X}(\mathsf{F}\,\varphi))))$$
$$\therefore\ \Psi, (\omega : (\neg(\mathsf{F}\,\varphi)))$$

36. $\mathsf{F}_\delta\, i$

$$\Psi, (\omega : \varphi)$$
$$\therefore\ \Psi, (\omega : (\mathsf{F}_{\delta_1}\,\varphi))$$

se $\delta_1 <_\iota \tau$,

37. $\mathsf{F}_\delta\, ii$

$$\Psi, (\omega : (\mathsf{X}(\mathsf{F}_{\delta_1 -_\iota \tau}\,\varphi)))$$
$$\therefore\ \Psi, (\omega : (\mathsf{F}_{\delta_1}\,\varphi))$$

se $\tau \leq_\iota \delta_1$,

38. $\neg\,\mathsf{F}_\delta\, i$

$$\Psi, (\omega : (\neg\,\varphi))$$
$$\therefore\ \Psi, (\omega : (\neg(\mathsf{F}_{\delta_1}\,\varphi)))$$

se $\delta_1 <_\iota \tau$,

39. $\neg\,\mathsf{F}_\delta\, ii$

$$\Psi, (\omega : (\neg(\mathsf{X}(\mathsf{F}_{\delta_1 -_\iota \tau}\,\varphi))))$$
$$\therefore\ \Psi, (\omega : (\neg(\mathsf{F}_{\delta_1}\,\varphi)))$$

se $\tau \leq_\iota \delta_1$,

40. $\mathsf{F}_{\preceq\delta}\, i$

$$\Psi, (\omega : \varphi)$$
$$\therefore\ \Psi, (\omega : (\mathsf{F}_{\preceq\delta_1}\, \varphi))$$

se $\delta_1 <_\iota \tau$,

41. $\mathsf{F}_{\preceq\delta}\, ii$

$$\Psi, (\omega : \varphi)\ ;$$
$$\Psi, (\omega : (\neg\, \varphi)), (\omega : (\mathsf{X}(\mathsf{F}_{\preceq\delta_1 -_\iota \tau}\, \varphi)))$$
$$\therefore\ \Psi, (\omega : (\mathsf{F}_{\preceq\delta_1}\, \varphi))$$

se $\tau \leq_\iota \delta_1$,

42. $\neg\, \mathsf{F}_{\preceq\delta}\, i$

$$\Psi, (\omega : (\neg\, \varphi))$$
$$\therefore\ \Psi, (\omega : (\neg(\mathsf{F}_{\preceq\delta_1}\, \varphi)))$$

se $\delta_1 <_\iota \tau$,

43. $\neg\, \mathsf{F}_{\preceq\delta}\, ii$

$$\Psi, (\omega : (\neg\, \varphi)), (\omega : (\neg(\mathsf{X}(\mathsf{F}_{\preceq\delta_1 -_\iota \tau}\, \varphi))))$$
$$\therefore\ \Psi, (\omega : (\neg(\mathsf{F}_{\preceq\delta_1}\, \varphi)))$$

se $\delta_1 \leq_\iota \tau$,

44. G

$$\Psi, (\omega : \varphi), (\omega : (\neg(\mathsf{X}(\neg(\mathsf{G}\, \varphi)))))$$
$$\therefore\ \Psi, (\omega : (\mathsf{G}\, \varphi))$$

45. $\neg\, \mathsf{G}$

$$\Psi, (\omega : (\neg\, \varphi))\ ;$$
$$\Psi, (\omega : \varphi), (\omega : (\mathsf{X}(\neg(\mathsf{G}\, \varphi))))$$
$$\therefore\ \Psi, (\omega : (\neg(\mathsf{G}\, \varphi))).$$

46. $\mathsf{G}_\delta\, i$

$$\Psi, (\omega : \varphi)$$
$$\therefore\ \Psi, (\omega : (\mathsf{G}_{\delta_1}\, \varphi))$$

se $\delta_1 <_\iota \tau$,

47. $\mathsf{G}_\delta\, ii$

$$\Psi, (\omega : (\neg(\mathsf{X}(\neg(\mathsf{G}_{\delta_1 -_\iota \tau}\, \varphi)))))$$
$$\therefore\ \Psi, (\omega : (\mathsf{G}_{\delta_1}\, \varphi))$$

se $\tau \leq_\iota \delta_1$,

48. $\neg\, \mathsf{G}_{\delta}\, i$

$$\Psi, (\omega : (\neg\, \varphi))$$
$$\therefore\; \Psi, (\omega : (\neg(\mathsf{G}_{\delta_1}\, \varphi)))$$

se $\delta_1 <_\iota \tau$,

49. $\neg\, \mathsf{G}_{\delta}\, ii$

$$\Psi, (\omega : (\mathsf{X}(\neg(\mathsf{G}_{\delta_1 -_\iota \tau}\, \varphi))))$$
$$\therefore\; \Psi, (\omega : (\neg(\mathsf{G}_{\delta_1}\, \varphi)))$$

se $\tau \leq_\iota \delta_1$,

50. $\mathsf{G}_{\preceq\delta}\, i$

$$\Psi, (\omega : \varphi)$$
$$\therefore\; \Psi, (\omega : (\mathsf{G}_{\preceq\delta_1}\, \varphi))$$

se $\delta_1 <_\iota \tau$,

51. $\mathsf{G}_{\preceq\delta}\, ii$

$$\Psi, (\omega : \varphi), (\omega : (\neg(\mathsf{X}(\neg(\mathsf{G}_{\preceq\delta_1 -_\iota \tau}\, \varphi)))))$$
$$\therefore\; \Psi, (\omega : (\mathsf{G}_{\preceq\delta_1}\, \varphi))$$

se $\delta_1 \leq_\iota \tau$,

52. $\neg\, \mathsf{G}_{\preceq\delta}\, i$

$$\Psi, (\omega : (\neg\, \varphi))$$
$$\therefore\; \Psi, (\omega : (\neg(\mathsf{G}_{\preceq\delta_1}\, \varphi)))$$

se $\delta_1 <_\iota \tau$,

53. $\neg\, \mathsf{G}_{\preceq\delta}\, ii$

$$\Psi, (\omega : (\neg\, \varphi))\ ;$$
$$\Psi, (\omega : \varphi), (\omega : (\mathsf{X}(\neg(\mathsf{G}_{\preceq\delta_1 -_\iota \tau}\, \varphi))))$$
$$\therefore\; \Psi, (\omega : (\neg(\mathsf{G}_{\preceq\delta_1}\, \varphi)))$$

se $\tau \leq_\iota \delta_1$,

54. S

$$\Psi, (\omega : \varphi_2)\ ;$$
$$\Psi, (\neg(\omega : \varphi_2)), (\omega : \varphi_1), (\omega : (\mathsf{Y}(\varphi_1\, \mathsf{S}\, \varphi_2)))$$
$$\therefore\; \Psi, (\omega : (\varphi_1\, \mathsf{S}\, \varphi_2))$$

55. $\neg\,\mathsf{S}$

$$\Psi,(\omega:(\neg\,\varphi_1)),(\omega:(\neg\,\varphi_2))\ ;$$
$$\Psi,(\omega:(\neg\,\varphi_2)),(\omega:(\neg(\mathsf{Y}(\varphi_1\,\mathsf{S}\,\varphi_2))))$$
$$\therefore\ \Psi,(\omega:(\neg(\varphi_1\,\mathsf{S}\,\varphi_2)))$$

56. P

$$\Psi,(\omega:\varphi)\ ;$$
$$\Psi,(\omega:(\neg\,\varphi)),(\omega:(\mathsf{Y}(\mathsf{P}\,\varphi)))$$
$$\therefore\ \Psi,(\omega:(\mathsf{P}\,\varphi))$$

57. $\neg\,\mathsf{P}$

$$\Psi,(\omega:(\neg\,\varphi)),(\omega:(\neg(\mathsf{Y}(\mathsf{P}\,\varphi))))$$
$$\therefore\ \Psi,(\omega:(\neg(\mathsf{P}\,\varphi)))$$

58. $\mathsf{P}_\delta\,i$

$$\Psi,(\omega:\varphi)$$
$$\therefore\ \Psi,(\omega:(\mathsf{P}_{\delta_1}\,\varphi))$$

se $\delta_1 <_\iota \tau$,

59. $\mathsf{P}_\delta\,ii$

$$\Psi,(\omega:(\mathsf{Y}(\mathsf{P}_{\delta_1 -_\iota \tau}\,\varphi))$$
$$\therefore\ \Psi,(\omega:(\mathsf{P}_\delta\,\varphi))$$

se $\tau \leq_\iota \delta_1$,

60. $\neg\,\mathsf{P}_\delta\,i$

$$\Psi,(\omega:(\neg\,\varphi))$$
$$\therefore\ \Psi,(\omega:(\neg(\mathsf{P}_{\delta_1}\,\varphi)))$$

se $\delta_1 <_\iota \tau$,

61. $\neg\,\mathsf{P}_\delta\,ii$

$$\Psi,(\omega:(\neg(\mathsf{Y}(\varphi_1\,\mathsf{P}_{\delta_1 -_\iota \tau}\,\varphi))))$$
$$\therefore\ \Psi,(\omega:(\neg(\mathsf{P}_{\delta_1}\,\varphi)))$$

se $\tau \leq_\iota \delta_1$,

62. $\mathsf{P}_{\preceq\delta}\,i$

$$\Psi,(\omega:\varphi)$$
$$\therefore\ \Psi,(\omega:(\mathsf{P}_{\preceq\delta_1}\,\varphi))$$

se $\delta_1 <_\iota \tau$,

63. $\mathsf{P}_{\preceq\delta}\, ii$

$\Psi, (\omega : \varphi)\ ;$

$\Psi, (\omega : (\neg\, \varphi)), (\omega : (\mathsf{P}_{\preceq\delta_1 -_\iota \tau}\, \varphi))$

$\therefore\ \Psi, (\omega : (\mathsf{P}_{\preceq\delta_1}\, \varphi))$

se $\tau \leq_\iota \delta_1$,

64. $\neg\, \mathsf{P}_{\preceq\delta}\, i$

$\Psi, (\omega : (\neg\, \varphi))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{P}_{\preceq\delta_1}\, \varphi)))$

se $\delta_1 <_\iota \tau$,

65. $\neg\, \mathsf{P}_{\preceq\delta}\, ii$

$\Psi, (\omega : (\neg\, \varphi)), (\omega : (\neg(\mathsf{Y}(\mathsf{P}_{\preceq\delta_1 -_\iota \tau}\, \varphi))))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{P}_{\preceq\delta_1}\, \varphi)))$

se $\delta_1 \leq_\iota \tau$,

66. H

$\Psi, (\omega : \varphi), (\omega : (\neg(\mathsf{Y}(\neg(\mathsf{H}\, \varphi)))))$

$\therefore\ \Psi, (\omega : (\mathsf{H}\, \varphi))$

67. $\neg\, \mathsf{H}$

$\Psi, (\omega : (\neg\, \varphi))\ ;$

$\Psi, (\omega : \varphi), (\omega : (\mathsf{Y}(\neg(\mathsf{H}\, \varphi))))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{H}\, \varphi)))$.

68. $\mathsf{H}_\delta\, i$

$\Psi, (\omega : \varphi)$

$\therefore\ \Psi, (\omega : (\mathsf{H}_{\delta_1}\, \varphi))$

se $\delta_1 <_\iota \tau$,

69. $\mathsf{H}_\delta\, ii$

$\Psi, (\omega : (\neg(\mathsf{Y}(\neg(\mathsf{H}_{\delta_1 -_\iota \tau}\, \varphi)))))$

$\therefore\ \Psi, (\omega : (\mathsf{H}_{\delta_1}\, \varphi))$

se $\tau \leq_\iota \delta_1$,

70. $\neg\, \mathsf{H}_\delta\, i$

$\Psi, (\omega : (\neg\, \varphi))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{H}_{\delta_1}\, \varphi)))$

se $\delta_1 <_\iota \tau$,

71. $\neg\, \mathsf{H}_{\delta}\, ii$

$\Psi, (\omega : (\mathsf{Y}(\neg(\mathsf{H}_{\delta_1 -_\iota \tau}\, \varphi)))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{H}_{\delta_1}\, \varphi)))$

se $\tau \leq_\iota \delta_1$,

72. $\mathsf{H}_{\preceq \delta}\, i$

$\Psi, (\omega : \varphi)$

$\therefore\ \Psi, (\omega : (\mathsf{H}_{\preceq \delta_1}\, \varphi))$

se $\delta_1 <_\iota \tau$,

73. $\mathsf{H}_{\preceq \delta}\, ii$

$\Psi, (\omega : \varphi), (\omega : (\neg(\mathsf{Y}(\neg(\mathsf{H}_{\preceq \delta_1 -_\iota \tau}\, \varphi)))))$

$\therefore\ \Psi, (\omega : (\mathsf{H}_{\preceq \delta_1}\, \varphi))$

se $\delta_1 \leq_\iota \tau$,

74. $\neg\, \mathsf{H}_{\preceq \delta}\, i$

$\Psi, (\omega : (\neg\, \varphi))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{H}_{\preceq \delta_1}\, \varphi)))$

se $\delta_1 <_\iota \tau$,

75. $\neg\, \mathsf{H}_{\preceq \delta}\, ii$

$\Psi, (\omega : (\neg\, \varphi))$;

$\Psi, (\omega : \varphi), (\omega : (\mathsf{Y}(\neg(\mathsf{H}_{\preceq \delta_1 -_\iota \tau}\, \varphi))))$

$\therefore\ \Psi, (\omega : (\neg(\mathsf{H}_{\preceq \delta_1}\, \varphi)))$

se $\tau \leq_\iota \delta_1$.

□

Um conjunto de fórmulas poderá ser um conjunto fechado para uma regra de inferência ou um conjunto esgotado. Estas noções são apresentadas em seguida.

Definição 4.3.4 (Conjunto Fechado para uma Regra de RI_Γ)
Um subconjunto Ψ de F_Γ diz-se fechado para uma regra ϱ de RI_Γ sse, se Ψ contém a conclusão da regra, então Ψ contém uma das suas premissas.

□

Definição 4.3.5 (Conjunto Esgotado)
Um subconjunto Ψ de F_Γ diz-se esgotado sse:

- Ψ contém um absurdo

  ou

- Ψ é fechado para todas as regras de inferência de RI_Γ.

□

Introduzimos a seguinte notação sobre regras de inferência:

Notação 4.3.6 Seja ϱ uma regra de inferência de RI_Γ. Designamos por

- $conc(\varrho)$ a conclusão da regra ϱ,
- $perm(\varrho)$ a premissa da regra ϱ, se esta é uma regra unária,
- $perm_1(\varrho), \ldots, perm_n(\varrho)$ as premissas da regra ϱ, se esta é uma regra com aridade n.

□

A estrutura de um tableau é uma árvore com ramificação finita, isto é, a partir de cada nó gera-se um número finito de nós. A este tipo de árvore dá-se o nome de árvore finitamente gerada.

Definição 4.3.7 (Árvore Finitamente Gerada)
Uma árvore finitamente gerada Avr é um conjunto de sequências de naturais que verifica as seguintes propriedades:

- Avr é fechada para prefixos, isto é, se a sequência $\eta.n$ pertence a Avr para $n \in I\!N$, então η pertence a Avr,
- se a sequência $\eta.n$ pertence a Avr para para $n \in I\!N$, então as sequências $\eta.i$ para $1 \leq i \leq n$ também pertencem a Avr.

A sequência vazia ε diz-se a *raiz* da árvore. Cada sequência do conjunto diz-se um *nó* da árvore.

□

Introduzimos a seguinte nomenclatura para trabalhar com árvores:

Notação 4.3.8 Seja Avr uma árvore finitamente gerada.

- Diz-se que o nó η de Avr é *maximal* sse não é prefixo de nenhum outro nó em Arv.
- Um *ramo* β de Avr é um subconjunto de Avr tal que:

  - β é fechada para prefixos;
  - se os nós $\eta.i, \eta.j \in \beta$, então $i = j$;
  - se o nó $\eta \in \beta$ não é prefixo de nenhum outro nó em β, então η é maximal em Avr.

- Seja β um ramo finito; η_β representa o *nó maximal* que se encontra no ramo β.
- $ramos(Avr)$ representa o conjunto dos ramos da árvore Avr.

□

Um tableau é constituído por uma árvore finitamente gerada com uma etiquetação dos seus nós. No entanto, uma especificação de objecto contém acções e estas por sua vez apresentam um tempo de execução. Quando se pretende verificar se uma fórmula é consequência de uma especificação de um objecto, é necessário ter em conta a duração da execução das acções, o que não se encontra implícito na definição do sistema de tableaux. Para ultrapassar este problema, e sempre que se pretende verificar uma relação de derivação, será necessário acrescentar fórmulas temporais à raiz do tableau de modo a garantir que cada acção respeite a duração especificada na assinatura. O conjunto acrescentado à raiz tem o nome de *fecho de uma assinatura*.

Definição 4.3.9 (Fecho de uma Assinatura)
O fecho de uma assinatura de objecto $\Sigma = \langle \Delta, \tau, GACT, GATT \rangle$ é o menor subconjunto Υ^Σ de F_Σ que verifica as seguintes condições:

- para todo o par $(att_1, att_2) \in ANM$ e toda a acção $act \in ACT$ tais que
  - $par(act) = \emptyset$,
  - $act \neq \bot$,
  - $act \in res(att_1)$,

  temos:

  - $(\mathsf{G}((att_1 = act) \Leftrightarrow (att_2 = act))) \in \Upsilon^\Sigma$
    se $dur(act) = 0$,
  - $((\mathsf{G}((att_1 = act) \Rightarrow ((att_2 = \bot)\, \mathsf{U}_{dur(act)}\, (att_2 = act))))\wedge$
    $(\mathsf{G}((att_2 = act) \Rightarrow ((att_1 = \bot)\, \mathsf{S}_{dur(act)}\, (att_1 = act))))) \in \Upsilon^\Sigma$
    se $dur(act) \neq 0$;

- para todo o par $(att_1, att_2) \in ANM$ e toda a acção $act \in ACT$ tais que
  - $par(act) \neq \emptyset$,

- $act(x) \in res(att_1)$ para alguma variável $x \in X_{par(act)}$,

temos:

- $(\mathsf{G}((att_1 = act(x)) \Leftrightarrow (att_2 = act(x)))) \in \Upsilon^\Sigma$
  se $dur(act) = 0$,
- $((\mathsf{G}((att_1 = act(x)) \Rightarrow ((att_2 = \perp) \, \mathsf{U}_{dur(act)} \, (att_2 = act(x)))))\wedge$
  $(\mathsf{G}((att_2 = act(x)) \Rightarrow ((att_1 = \perp) \, \mathsf{S}_{dur(act)} \, (att_1 = act(x))))) \in \Upsilon^\Sigma$
  se $dur(act) \neq 0$.

□

Definição 4.3.10 (Tableau)
Seja Ψ um conjunto finito de F_Γ tal que $pref(\Psi) = \{\omega\}$. Um tableau com raiz Ψ é um par $\mathcal{T} = \langle Avr, et\rangle$ onde:

- Avr é uma árvore finitamente gerada,
- $et : Avr \to 2^{F_\Gamma}$ é uma aplicação em que $et(\varepsilon) = \Psi \cup \Upsilon^\Sigma_\omega$,
- para qualquer nó η de Avr verifica-se que:
  - η um nó maximal de Avr sse $et(\eta)$ é um conjunto esgotado,
  - se o nó η não é maximal, então existe uma regra de inferência ϱ de RI_Γ com aridade n e nós distintos $\eta.1, \ldots, \eta.n$ em Avr tal que:
    * $conc(\varrho) = et(\eta)$,
    * $perm_i(\varrho) = et(\eta.i)$ para $i = 1, \ldots, n$,
    * $\eta.(n+1)$ não é um nó de Avr.

□

É possível estender a aplicação et de modo a interpretar os ramos de uma árvore. Seja β um ramo de Arv, o valor da função et aplicado a β é calculado do seguinte modo:

$$et(\beta) = \bigcup_{\eta\in\beta} et(\eta).$$

Notação 4.3.11 Sejam ψ uma fórmula de F_Γ, Ψ um conjunto de fórmulas, η um nó e β um ramo. No seguimento será usada a seguinte notação:

- $\psi \in \eta$ sse $\psi \in et(\eta)$,
- $\Psi \subset \eta$ sse $\Psi \subseteq et(\eta)$,

- $\psi \in \beta$ sse $\psi \in et(\beta)$,
- $\Psi \subset \beta$ sse $\Psi \subseteq et(\beta)$.

□

Em seguida apresentamos algumas definições relacionadas com tableaux.

Definição 4.3.12 (Ramo Fechado e Aberto)
Uma ramo β de um tableau diz-se fechado sse contém um absurdo. Caso contrário, o ramo diz-se aberto.

□

Definição 4.3.13 (Confutação e Confirmação)
Um tableau com raiz Ψ é uma confutação sse todos os seus ramos são fechados. Caso contrário, o tableau é uma confirmação.

□

Definição 4.3.14 (Relação de Derivação)
Sejam Φ um subconjunto finito de F_Σ, φ uma fórmula de F_Σ e ω um prefixo de Ω. Diz-se que φ é derivável a partir de Φ, escrevendo-se $\Phi \vdash_\Sigma \varphi$, sse o tableau com raiz $\Phi_\omega \cup \{(\neg\, \varphi)_\omega\}$ é uma confutação.

□

É também usual designar a relação de derivação por consequência sintáctica de uma fórmula por um conjunto de fórmulas.

Analogamente às consequências semânticas apresentadas anteriormente, iremos indicar quando é que uma fórmula é consequência sintáctica de uma especificação de objecto. No entanto, como a noção de assinatura de prefixos não se encontra definida numa especificação de objecto, a consequência sintáctica será a nível da assinatura de objecto, utilizando uma assinatura de prefixos.

Definição 4.3.15 (Consequência Sintáctica de uma Especificação de Objecto)
Sejam $Ob = \langle \Sigma, Ax \rangle$ uma especificação de objecto e φ um fórmula de F_Σ. Diz-se que φ é consequência sintáctica da especificação Ob, escrevendo-se $Ob \vdash_\Sigma \varphi$, sse $Ax \vdash_\Sigma \varphi$.

□

4.4 Decidibilidade

Nesta secção iremos verificar que o sistema de tableaux apresentado é decidível. Primeiro provamos que é correcto e completo, adoptando a sequência de raciocínio apresentada em [Fitting 83]. Em seguida definimos um algoritmo para tirar conclusões de um tableau em tempo finito, mesmo que este apresente ramos infinitos. Neste último ponto, iremos recorrer à noção de ciclo e de intervalo de influência de uma fórmula.

4.4.1 Correcção e Completude

A prova da correcção e da completude do sistema de tableaux irá ser feita em dois passos: primeiro verifica-se que um ramo de um tableau contém um absurdo sse contém um conjunto contraditório; de seguida verificamos que as regras de inferência são correctas para conjuntos contraditórios. Este processo de verificar a correcção e a completude de um sistema dedutivo é muito semelhante ao processo apresentado em [Fitting 83]. A principal diferença consiste em considerar ramos em vez de nós para o primeiro passo. A razão pela qual é necessário considerar ramos deve-se aos segundo e terceiro absurdos da definição 4.3.1, que só são detectáveis se analisarmos todo o ramo.

Proposição 4.4.1 (Absurdo vs Contraditório)
Sejam Ψ um subconjunto de F_Γ e β um ramo do tableau com raiz Ψ. O ramo β contém um absurdo sse β contém um conjunto contraditório.

Demonstração:
Verifiquemos separadamente ambos os sentidos da equivalência:

$\rightarrow$ Comecemos por verificar a seguinte implicação: se um conjunto Ψ de F_Γ contém um absurdo, então é contraditório. Supomos que o conjunto Ψ contém um absurdo, mas que existe uma interpretação de prefixos $I_p = \langle M, S\rangle$ que possibilite o conjunto. Por definição de absurdo, o conjunto Ψ verifica um dos seguintes casos:

- $\{(\omega : \varphi), (\omega : (\neg\, \varphi))\} \subseteq \Psi$

  Este caso consiste em dizer que $I \Vdash_\Gamma (\omega : \varphi)$ e $I_p \Vdash_\Gamma (\omega : (\neg\, \varphi))$, o que é impossível uma vez que uma interpretação não pode satisfazer uma fórmula e a negação dessa fórmula no mesmo estado;

- $(\omega : (\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2)) \in \Psi$ e para qualquer natural n existe um prefixo $w' \in \Omega$ tal que $\{(\omega\alpha^n\omega'), (\omega' : (\neg\, \varphi_2))\} \subseteq \Psi)$, em que $\varphi_1, \varphi_2 \in F_\Sigma$ e $\omega \in \Omega$

  Supondo que existe uma interpretação de prefixos $I_p = \langle K, S\rangle$ em que $I_p \Vdash_\Gamma (\omega : (\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2))$, então existe um estado w tal que $(S(\omega), w) \in R^*$ e $\langle K, w\rangle \Vdash_\Sigma \varphi_2$. Vamos supor que $(S(\omega), w) \in R^n$ para um natural n. Tendo em conta a definição do segundo caso, existe um prefixo ω' tal que $I_p \Vdash_\Gamma (\omega\alpha^n\omega')$ e $I_p \Vdash_\Gamma (\omega' : (\neg(\varphi_2)))$. Isto é, $(S(\omega), S(\omega')) \in R^n$ e $\langle K, S(\omega')\rangle \Vdash_\Sigma (\neg\, \varphi_2)$. Como a relação R é linear, então os estados w e $S(\omega')$ são idênticos, o que origina uma impossibilidade;

- $\omega \in pref(\Psi)$ e para qualquer natural n existe um prefixo ω' tal que $(\omega'\alpha^n\omega) \in \Psi$

  Suponhamos igualmente que existe uma interpretação de prefixos $I_p = \langle K, S\rangle$ em que $I_p \Vdash_\Gamma \Psi$. Como para qualquer natural n existem prefixos $\omega_0, \ldots, \omega_n$ tais que $\omega_0 = \omega', \omega_n = \omega$ e $\{(\omega_0\alpha\omega_1), \ldots, (\omega_{n-1}\alpha\omega_n)\} \subseteq \Psi$, temos que $I_p \Vdash_\Gamma (\omega_i\alpha\omega_{i+1})$ para todo $i = 0, \ldots, (n-1)$. Recorrendo à definição 4.2.7 de relação de satisfação para fórmulas prefixadas, temos que $(S(\omega_i), S(\omega_{i+1})) \in R$ para todo $i = 0, \ldots, (n-1)$. Existe portanto uma sequência infinita de estados na relação R no passado, o que entra em contradição com a definição 3.3.1 de estrutura de Kripke.

Como todo o conjunto que contém um absurdo é contraditório, se um ramo contém um absurdo então também contém um conjunto contraditório.

$\leftarrow$ Para este caso contruimos uma interpretação de prefixos $I_p = \langle\langle W, R, V\rangle, S\rangle$ tal que $I_p \Vdash_\Gamma \psi$ para qualquer fórmula $\psi \in \beta$.

- $W = \Omega$,
- $R = \{(\omega_1, \omega_2) \ : \ (\omega_1\alpha\omega_2) \in \beta\}$,
- $V_{att}(\omega) = \begin{cases} t & \text{se } (\omega : (att = t)) \in \beta, \\ \perp & \text{se } (\omega : (att = t)) \notin \beta \text{ para qualquer } t \in T_{\Sigma, res(att)} \\ & \text{ e } (att, att_1) \in ANM \text{ ou } (att_1, att) \in ANM \\ \text{indefinido} & \text{caso contrário} \end{cases}$
- $S = id_\Omega$, isto é, $S(\omega) = \omega$.

Como um ramo de um tableau contém a raiz e a raiz é um conjunto fechado para a assinatura Σ, o tempo de duração das acções é respeitado pela estrutura de Kripke $\langle W, R, V\rangle$. Prova-se facilmente por indução na estrutura de ψ que, se $\psi \in \beta$, então $I_p \Vdash_\Sigma \psi$.

□

Proposição 4.4.2 (Correcção das Regras de Inferência)
As regras de inferência de RI_Γ são correctas para conjuntos contraditórios, isto é, se as premissas de uma regra são conjuntos contraditórios então a conclusão é um conjunto contraditório.

Demonstração:
Provamos a correcção de algumas regras de inferência por contra-recíproco:

- $\Rightarrow$

  Seja $I_p = \langle K, S\rangle$ uma interpretação de prefixos tal que $I_p \Vdash_\Gamma \Psi$ e $I_p \Vdash_\Gamma (\omega : (\varphi_1 \Rightarrow \varphi_2))$. Por definição 4.2.7 podemos obter $\langle K, S(\omega)\rangle \Vdash_\Sigma (\varphi_1 \Rightarrow \varphi_2)$ e, por definição de satisfação 3.3.12 e de relação de satisfação 3.3.9, deduz-se que $\langle K, S(\omega)\rangle, \rho \Vdash_\Sigma (\neg\, \varphi_1)$ ou $\langle K, S(\omega)\rangle, \rho \Vdash_\Sigma \varphi_2$ para qualquer atribuição de variáveis ρ. Recorrendo novamente à definição 4.2.7, temos que $I_p \Vdash_\Gamma (\omega : (\neg\, \varphi_1))$ ou $I_p \Vdash_\Gamma (\omega : \varphi_2)$. Dada a satisfação do conjunto Ψ, podemos concluir que I_p satisfaz uma das premissas da regra $\Rightarrow$.

- $\forall$

  Seja $I_p = \langle K, \Omega\rangle$ uma interpretação de prefixos tal que $I_p \Vdash_\Gamma \Psi$ e $I_p \Vdash_\Gamma (\omega : (\forall x\ \varphi))$. Por definição 4.2.7 temos que $\langle K, S(\omega)\rangle \Vdash_\Sigma (\forall x\ \varphi)$. Por definição 3.3.12 de satisfação obtém-se que $\langle K, S(\omega)\rangle, \rho \Vdash_\Sigma (\forall x\ \varphi)$ para qualquer atribuição a variáveis ρ. Por definição 3.3.9 de relação de satisfação temos que $\langle K, S(\omega)\rangle, \rho' \Vdash_\Sigma \varphi$ para qualquer atribuição de variáveis $\rho' \approx_x \rho$. Recorrendo à proposição 3.3.11 podemos deduzir que $\langle K, S(\omega)\rangle, \rho \Vdash_\Sigma [\varphi]^x_{d_i}$ para qualquer elemento d_i de D. Recorrendo novamente às definições 3.3.12 e 4.2.7 temos que $I_p \Vdash_\Gamma (\omega : [\varphi]^x_{d_i})$ para qualquer $d_i \in D$.

- $\mathsf{X}\,i$

  Seja $I_p = \langle K, \Omega\rangle$ uma interpretação de prefixos tal que $I_p \Vdash_\Gamma \Psi$ e $I_p \Vdash_\Gamma (\omega_1 : (\mathsf{X}\,\varphi))$. Pelas definições 3.3.12 e 4.2.7 temos que $\langle K, S(\omega_1)\rangle \Vdash_\Sigma (\mathsf{X}\,\varphi)$, isto é, existe um estado w tal que $(S(\omega_1), w) \in R$ e $\langle K, w\rangle \Vdash_\Sigma \varphi$. Seja ω_2 um prefixo de Ω tal que $S(\omega_2) = w$; tem-se que $\langle K, S(\omega_2)\rangle \Vdash_\Sigma \varphi$. Pela definição 4.2.7, obtém-se que $I_p \Vdash_\Gamma (\omega_2 : \varphi)$.

- $\mathsf{U}_\delta ii$

  Seja $I_p = \langle K, \Omega\rangle$ uma interpretação de prefixos tal que $I_p \Vdash_\Gamma \Psi$, $I_p \Vdash_\Gamma \Vdash_\Sigma (\omega : (\varphi_1 \,\mathsf{U}_{\delta_1}\, \varphi_2))$. Pela definição 4.2.7 temos que $\langle K, S(\omega)\rangle \Vdash_\Sigma (\varphi_1 \,\mathsf{U}_{\delta_1}\, \varphi_2)$. Pelas definições 3.3.12 e 3.3.9 podemos deduzir que existe um estado w_1 visível a partir de $S(\omega)$ a uma distância inferior ou igual a δ_1 tal que que $I, w_1 \Vdash_\Sigma \varphi_2$ e $I, w_2 \Vdash_\Sigma \varphi_1$ para todo o estado w_2 entre $S(\omega)$ e w_1 na relação R^* distinto de w_1. Seja w_3 um estado de W em que $(S(\omega), w_3) \in R$. Como $m_K(S(\omega), w_1) \leq_\iota \delta_1$ podemos verificar que $m_K(w_3, w_1) = m_K(S(\omega), w_1) -_\iota \tau \ \leq_\iota \ \delta_1 -_\iota \tau$ e portanto verifica-se que $\langle K, w_3\rangle \Vdash_\Sigma (\varphi_1 \,\mathsf{U}_{\delta_1 -_\iota \tau}\, \varphi_2)$. Isto é, $\langle K, S(\omega)\rangle \Vdash_\Sigma (\mathsf{X}(\varphi_1 \,\mathsf{U}_{\delta_1 -_\iota \tau}\, \varphi_2))$. Pela definição 4.2.7 temos que $I_p \Vdash_\Sigma (\omega : \varphi_1)$ e que $I_p \Vdash_\Sigma (\mathsf{X}(\varphi_1 \,\mathsf{U}_{\delta_1 -_\iota \tau}\, \varphi_2))$.

$\square$

As duas proposições anteriores formam a base para provar a correcção e completude do sistema de tableaux. Em seguida enunciamos uma proposição que irá recorrer às duas anteriores e a partir da qual facilmente se prova a correcção e completude.

Proposição 4.4.3 (Confutação vs Contraditório)
Seja Ψ um subconjunto finito de F_Γ. Um tableau com raiz Ψ é uma confutação sse Ψ é um conjunto contraditório.

Demonstração:
Se o tableau é uma confirmação, então contém um ramo aberto β. Por definição de ramo aberto e pela proposição 4.4.1, β não pode conter um conjunto contraditório. Como a raiz do tableau está contida no ramo, podemos concluir que a raiz não é um conjunto contraditório. Fica assim provado um dos sentidos da equivalência. No outro sentido (se um tableau com raiz Ψ é uma confutação, então Ψ é um conjunto contraditório) consideramos dois tipos de tableaux:

Tableau finito: Neste caso efectuamos uma prova por indução na profundidade do tableau.

Base, $prof = 0$: Um tableau com profundidade 0 é confutado se a raiz é um absurdo. Pela proposição 4.4.1 temos que a raiz do tableau é um conjunto contraditório.

Hipótese, $prof \leq n$: Se um tableau com raiz Ψ com profundidade inferior ou igual a n é uma confutação, então Ψ é um conjunto contraditório.

Tese, $prof = n+1$: Consideremos um tableau $\mathcal{T}$ com profundidade igual a $n+1$ e os subtableaux $\mathcal{T}_1, \ldots, \mathcal{T}_n$ de $\mathcal{T}$ que se obtêm por aplicação da regra de inferência à raiz de $\mathcal{T}$. Se $\mathcal{T}$ é uma confutação, então pela definição 4.3.13 todos os ramos de $\mathcal{T}$ são fechados. Consequentemente, todos os tableaux $\mathcal{T}_1, \ldots, \mathcal{T}_n$ são confutações. Por hipótese de indução, as suas raizes são conjuntos contraditórios. Pela correcção das regras de inferência (proposição 4.4.2), temos que o conjunto Ψ é contraditório.

Tableau infinito: Para este caso suponhamos, sem perda de generalidade, que existe um único ramo infinito β.
Sendo β um ramo infinito com um absurdo, podemos concluir que o absurdo é do

segundo tipo apresentado na definição 4.3.1. Seja η o nó do ramo β que contém a fórmula $(\omega : (\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2))$. Se este nó constituísse um conjunto possível, existiria um ramo finito aberto contendo o conjunto de fórmulas $\{(\omega\alpha^n\omega'), (\omega' : \varphi_2)\}$ para algum prefixo $\omega' \in \Omega$ e para algum natural $n \in I\!N_o$. Como tal é impossível por hipótese, o nó η forma um conjunto contraditório. Estando este nó a uma distância finita da raiz e sendo os restantes ramos fechados, podemos deduzir que a raiz do tableau é um conjunto contraditório, recorrendo à definição 4.4.2.

□

Proposição 4.4.4 (Correcção e Completude)
Sejam $\Phi \subseteq F_\Sigma$ um conjunto finito, $\varphi \in F_\Gamma$ e $\omega \in \Omega$. Verifica-se a seguinte equivalência:

$\Phi \vdash_\Sigma \varphi$ sse $\Phi_\omega \vDash_\Gamma \varphi_\omega$.

Demonstração:
Trivial utilizando a definição 4.3.14 e as proposições 4.4.3 e 4.2.13.

□

4.4.2 Algoritmo de Decisão

Nesta secção apresentamos um algoritmo para verificar em tempo finito se um tableau com raiz Ψ é uma confirmação ou uma confutação. A identificação de um ciclo num ramo será essencial como critério de paragem do algoritmo. Para definir esses ciclos, novos conceitos irão ser introduzidos, como por exemplo, o de intervalo de influência.

Para definir um ciclo é necessário não só caracterizar o conjunto de fórmulas que formam o ciclo, como também indicar em que posição, no tableau, se encontram as condições necessárias para definir um ciclo, não sendo necessário procurar um ciclo sempre que se aplica uma regra de inferência. A próxima definição caracteriza os nós que são potenciais elementos de um ciclo.

Definição 4.4.5 (Nó Limitado)
Sejam η um nó do tableau $\langle Avr, et\rangle$ e ω_i, ω_s dois prefixos de Ω. Diz-se que η se encontra limitado inferiormente por ω_i e superiormente por ω_s sse η satisfaz as seguintes condições:

- η não contém um absurdo,
- $\omega_i, \omega_s \in (pref \circ et)(\eta)$,
- se $(\omega_1\alpha\omega_2) \in \eta$ então $\omega_1 \neq \omega_s$ e $\omega_2 \neq \omega_i$,
- $et(\eta)$ encontra-se fechado para todas as regras de inferência de RI_Γ, excepto para as regras $\mathsf{X}\,i$ e $\mathsf{Y}\,i$.

□

Um nó η diz-se limitado superiormente pelo prefixo ω_s se o nó não contém nenhuma informação sobre os prefixos para o futuro de ω_s e as únicas fórmulas não atómicas com operadores para o futuro em η são do tipo $(\omega_s : (\mathsf{X}\,\varphi))$ e $(\omega_s : (\neg(\mathsf{X}\,\varphi)))$. O mesmo nó diz-se limitado inferiormente se verificar as condições anteriores mas considerando o passado e os operadores para o passado.

Os nós limitados irão caracterizar os ciclos. No entanto, existem outros critérios para a definição do ciclo. Um desses critérios tem em conta as fórmulas existentes num nó limitado. O outro envolve a distância em tempo que separa os estados representados pelos prefixos que limitam os nós.

Definição 4.4.6 (Aplicações $\boldsymbol{form}, \boldsymbol{form_{\mathcal{F}}}$ e $\boldsymbol{form_{\mathcal{P}}}$)
Sejam Ψ um subconjunto de F_Γ e $\omega \in \Omega$. Definimos as seguintes aplicações:

- $form : 2^{F_\Gamma} \otimes \Gamma \to 2^{F_\Sigma}$ que, dado um conjunto de fórmulas e um prefixo, determina as fórmulas prefixadas pelo prefixo, é definida do seguinte modo:

$$form(\Psi, \omega) = \{\varphi \ : \ (\omega : \varphi) \in \Psi\},$$

- $form_{\mathcal{F}} : 2^{F_\Gamma} \otimes \Gamma \to 2^{F_\Sigma}$ que, dado um conjunto de fórmulas e um prefixo, determina as fórmulas no futuro imediato do prefixo, é definida do seguinte modo:

$$form_{\mathcal{F}}(\Psi, \omega) = \{\varphi \ : \ (\omega : (\mathsf{X}\,\varphi)) \in \Psi\} \cup \{(\neg\,\varphi) \ : \ (\omega : (\neg(\mathsf{X}\,\varphi))) \in \Psi\},$$

- $form_{\mathcal{P}} : 2^{F_\Gamma} \otimes \Gamma \to 2^{F_\Sigma}$ que, dado um conjunto de fórmulas e um prefixo, determina as fórmulas no passado imediato do prefixo, é definida do seguinte modo:

$$form_{\mathcal{P}}(\Psi, \omega) = \{\varphi \ : \ (\omega : (\mathsf{Y}\,\varphi)) \in \Psi\} \cup \{(\neg\,\varphi) \ : \ (\omega : (\neg(\mathsf{Y}\,\varphi))) \in \Psi\}.$$

□

Definição 4.4.7 (Intervalo de Influência)
Seja Σ uma assinatura com período τ. A função $\mathcal{V} : F_\Sigma \to I\!N_o$ que, dada uma fórmula, determina o intervalo de influência da mesma para o passado, tem a seguinte definição:

- $\mathcal{V}(\varphi) = 0$ se $\varphi \in A_\Sigma$,
- $\mathcal{V}(\neg\,\varphi) = \mathcal{V}(\varphi)$,
- $\mathcal{V}(\varphi_1 \Rightarrow \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\}$,
- $\mathcal{V}(\mathsf{X}\,\varphi) = \mathcal{V}(\varphi) - 1$,

- $\mathcal{V}(\varphi_1 \, \mathsf{U}_\delta \, \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\} - (\delta \div_\iota \tau)$,
- $\mathcal{V}(\varphi_1 \, \mathsf{U}_{\preceq\delta} \, \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\} - (\delta \div_\iota \tau)$
  se $\delta \neq \iota$,
- $\mathcal{V}(\varphi_1 \, \mathsf{U}_{\preceq\iota} \, \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\}$,
- $\mathcal{V}(\mathsf{Y} \, \varphi) = \mathcal{V}(\varphi) + 1$,
- $\mathcal{V}(\varphi_1 \, \mathsf{S}_\delta \, \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\} + (\delta \div_\iota \tau)$,
- $\mathcal{V}(\varphi_1 \, \mathsf{S}_{\preceq\delta} \, \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\} + (\delta \div_\iota \tau)$
  se $\delta \neq \iota$,
- $\mathcal{V}(\varphi_1 \, \mathsf{S}_{\preceq\iota} \, \varphi_2) = \max\{\mathcal{V}(\varphi_1), \mathcal{V}(\varphi_2)\}$.

$\square$

A subtracção utilizada é a subtracção natural, que apresenta o valor zero sempre que a subtracção inteira seja negativa.

O intervalo de influência de uma fórmula é um natural e representa o número de prefixos para o passado necessários para verificar essa mesma fórmula. Este cálculo tem em consideração que cada prefixo representa um estado e que dois estados consecutivos se distanciam de τ unidades de tempo. Para os operadores temporais para o futuro, é efectuada a subtracção natural do resultado obtido pela aplicação nas subfórmulas, com a divisão do valor de tempo da fórmula pelo período da assinatura. Para os operadores temporais para o passado é efectuada uma adição. No caso das fórmulas do tipo $(\varphi_1 \mathsf{U}_{\preceq\iota} \varphi_2)$, não se efectua qualquer tipo de subtracção porque não é conhecido exactamente em que estado se verifica a fórmula φ_2. Por defeito, nestes casos, consideramos que é o estado actual. Para as fórmulas do tipo $(\varphi_1 \, \mathsf{S}_{\preceq\iota} \, \varphi_2)$ é feito o mesmo raciocínio.

Notação 4.4.8 A extensão da função acima para conjuntos é feita do seguinte modo:

$$\mathcal{V}(\Phi) = \max\{\mathcal{V}(\varphi) : \varphi \in \Phi\}.$$

$\square$

Finalmente reunimos todas as definições necessárias para caracterizar um ciclo num ramo. Existem dois tipos de ciclos: ciclos para o futuro e ciclos para o passado.

Definição 4.4.9 (Ramo com Ciclo para o Futuro)
Seja $\mathcal{T}$ um tableau com raiz Ψ. Diz-se que o ramo β do tableau contém um ciclo para o futuro se contém nós η_1, η_2 e η_3 com limites superiores ν_1, ν_2 e ν_3, respectivamente, tais que:

- $(\nu_1 \alpha^p \nu_2) \in \beta$ para algum $p \in I\!N$,
- $form(\beta, \nu_1) = form(\beta, \nu_2)$,
- existe $n \in I\!N_o$ tal que
  - $(\mathcal{V} \circ form_{\mathcal{F}})(\beta, \nu_3) \leq n$,
  - $(\nu_2 \alpha^n \nu_3) \in \beta$.

$\square$

Uma ramo com um ciclo para o futuro é caraterizado por apresentar um conjunto de fórmulas prefixadas por dois prefixos distintos mas ligados por uma cadeia finita de prefixos na relação α. O terceiro nó da definição de ciclo permite garantir que as fórmulas que pertencem aos nós posteriores não irão alterar o conjunto das fórmulas prefixadas por ν_2. Portanto, todas as fórmulas relevantes para a definição do conjunto de fórmulas prefixadas por ν_2 encontram-se nos nós anteriores a η_3. No caso particular em que o valor calculado por $\mathcal{V} \circ form_{\mathcal{F}}(\beta, \nu_3)$ é zero, é possível escolher o valor zero para n e, consequentemente, fazer coincidir os nós η_2 e η_3. Neste caso, são necessários apenas dois nós distintos para definir um ciclo.

No caso dos ciclos para o passado, a sua definição é mais simplificada do que a definição anterior.

Definição 4.4.10 (Ramo com Ciclo para o Passado)
Seja $\mathcal{T}$ um tableau com raiz Ψ. Diz-se que o ramo β do tableau contém um ciclo para o passado se contém nós η_1 e η_2 com limites inferiores ℓ_1 e ℓ_2, respectivamente, tais que:

- $(\ell_1 \alpha^p \ell_2) \in \beta$ para algum $p \in I\!N$,
- $form_{\mathcal{P}}(\beta, \ell_1) = form_{\mathcal{P}}(\beta, \ell_2)$.

$\square$

A razão pela qual a definição de ciclo para o passado é mais simples do que a definição de ciclo para o futuro, está relacionada com as definições de absurdo para o futuro e para o passado apresentadas em 4.3.1. A definição de absurdo relacionada com o futuro é mais complexa do que a relacionada com o passado. Consequentemente, são necessárias menos condições para verificar o último absurdo. Esta diferença será relevante para a proposição 4.4.13, onde se estabelecerão definições equivalentes às de confutação e de confirmação de um tableaux.

É possível classificar um ciclo para o futuro em dois tipos: ciclo aberto e ciclo fechado.

Definição 4.4.11 (Ciclo Aberto e Ciclo Fechado)
Seja β um ramo com um ciclo para o futuro caracterizado pelos nós η_1, η_2 e η_3 com limites ν_1, ν_2 e ν_3, respectivamente. Diz-se que o ciclo é aberto se, para toda a fórmula $(\nu_1 : (\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2))$ em η_1, existe um prefixo ω tal que:

- $(\nu_1 \alpha^* \omega), (\omega \alpha^* \nu_2) \in \beta$,
- $(\omega : \varphi_2) \in \beta$.

Caso contrário diz-se que o ciclo é fechado.

□

A ideia por detrás da classificação de ciclo aberto ou fechado é semelhante à ideia por detrás da classificação de ramos abertos ou fechados. No entanto, uma ramo aberto poderá conter um ciclo fechado e um ramo fechado poderá conter um ciclo aberto. Um ciclo diz-se fechado se contém um potencial absurdo do segundo tipo apresentado em 4.3.1. Isto é, se contém uma fórmula do tipo $(\varphi_1 \mathsf{U}_{\preceq_\iota} \varphi_2)$ prefixada por ν_1 mas não contém a correspondente fórmula φ_2 prefixada por algum prefixo que represente um estado no futuro de ν_1 em η_3. Ou seja, para todo prefixo ω e natural n, se $(\nu_1 \alpha^n \omega) \in \eta_3$, então $(\omega : (\neg \varphi_2)) \in \beta$. Ramos que apresentam um ciclo fechado têm grandes potencialidades de serem ramos fechados. Caso exista um prefixo ω que não verifique as condições anteriores para todas as fórmulas com o operador temporal $\mathsf{U}_{\preceq_\iota}$, o ciclo diz-se aberto. De igual modo, um ramo com um ciclo aberto é um potencial ramo aberto. Este resultado é mais forte do que o anterior, porque é possível afirmar que um tableau com um ciclo aberto é uma confirmação.

No caso de ciclos para o passado não se justifica uma classificação análoga à anterior porque não faz sentido definir ciclo aberto para o passado. Utilizando a notação anterior, todo o ciclo para o passado é fechado.

De seguida apresentamos um resultado que se pode obter a partir de um ramo aberto que contém um ciclo fechado.

Proposição 4.4.12 Seja β_1 um ramo aberto com um ciclo fechado η_1, η_2 e η_3 onde ν_1, ν_2 e ν_3 são os respectivos limites. Sejam φ uma fórmula de F_Σ e ω_1 um prefixo. Se o conjunto $\{(\nu_2 \alpha^n \omega_2), (\omega_2 : \varphi)\}$ é subconjunto de β_1, então existe um prefixo ω_1 e existe um ramo aberto β_2 que contém o conjunto $\{(\nu_1 \alpha^n \omega_1), (\omega_1 : \varphi)\}$.

Demonstração:
A demonstração desta proposição é feita com base na família $\mathcal{B} = \{\mathcal{B}_i\}_{i \in \mathbb{N}_o}$ de conjuntos de ramos do tableau $\mathcal{T}$. Estes conjuntos são definidos indutivamente do seguinte modo:

- $\mathcal{B}_0 = \{\beta \in ramos(\mathcal{T}) : \ \eta_1 \in \beta\}$,
- $\mathcal{B}_{i+1} = \left\{\beta \in \mathcal{B}_i : \ \{(\nu_1 \alpha^{i+1} \omega), (\omega : \varphi)\} \subseteq \beta \text{ sse } \{(\nu_2 \alpha^{i+1} \omega'), (\omega' : \varphi)\} \subseteq \beta_1\right\}$.

Descrevemos de seguida duas propriedades verificadas pela família $\mathcal{B}$.

Lema 1 Para todo natural i, se o conjunto $\mathcal{B}_i$ não é vazio, então contém um ramo aberto.

Demonstração:
A prova é feita por indução sobre i.

Base, $i = 0$: Temos que β_1 é um ramo aberto e $\beta_1 \in \mathcal{B}_0$ uma vez que $\eta_1 \in \beta_1$.

Hipótese, $i \leq n$: Se $\mathcal{B}_i \neq \emptyset$, então existe um ramo aberto $\beta' \in \mathcal{B}_i$.

Passo, $i = n+1$: Supomos que o conjunto $\mathcal{B}_{n+1}$ não é vazio. Logo, o conjunto $\mathcal{B}_n$ também não é vazio e, por hipótese de indução, contém ramos abertos. Sejam $\mathcal{A}_n$ e $\mathcal{A}_{n+1}$ os conjuntos dos ramos abertos de $\mathcal{B}_n$ e $\mathcal{B}_{n+1}$, respectivamente. Verifica-se que $\mathcal{A}_{n+1} = \mathcal{A}_n \cap \mathcal{B}_{n+1}$, isto é, o conjunto $\mathcal{B}_{n+1}$ não contém ramos abertos que não se encontrem em $\mathcal{A}_n$. Se todos os ramos em $\mathcal{A}_n$ não pertencessem a $\mathcal{B}_{n+1}$, isto queria dizer que o conjunto das fórmulas prefixadas por ω_1 tal que $(\nu_1 \alpha^{n+1} \omega_1) \in \beta$ para $\beta \in \mathcal{B}_{n+1}$, era contraditório. Mas como este conjunto seria igual ao conjunto prefixado por ω_2 onde $(\nu_2 \alpha^{n+1} \omega_2) \in \beta_1$, podia-se concluir que o ramo β_1 era fechado (proposição 4.4.1), o que é uma contradição. Consequentemente, podemos concluir que existe um ramo em $\mathcal{B}_{n+1} \cap \mathcal{A}_n$.

□

Lema 2 Se para algum $n \in I\!N_o$ existe um prefixo ω tal que $(\nu_1 \alpha^n \omega) \in \beta_1$, então o conjunto $\mathcal{B}_n$ não é vazio.

Demonstração:
Por absurdo, supomos que existe um natural $i < n$ tal que $\mathcal{B}_i \neq \emptyset$, mas $\mathcal{B}_{i+1} = \emptyset$. Esta suposição só é possível se o conjunto das fórmulas prefixadas por ω, em que $(\nu_2 \alpha^i \omega) \in \beta_1$, é um conjunto contraditório, ou se o conjunto $\mathcal{B}_i$ não contém ramos abertos. Ambas as suposições são falsas, logo o conjunto $\mathcal{B}_{i+1}$ não é vazio.

□

A demonstração desta proposição é trivial se tivermos em conta os dois lemas anteriores. Se $\{(\nu_2 \alpha^n \omega_2), (\omega_2 : \varphi)\} \subseteq \beta_1$, então pelos lemas 1 e 2, o conjunto $\mathcal{B}_n$ não é vazio e contém pelos menos um ramo aberto. Seja β_2 um dos ramos abertos em $\mathcal{B}_n$. Por definição do conjunto $\mathcal{B}_n$, existe um prefixo ω_1 tal que $\{(\nu_1 \alpha^n \omega_1), (\omega_1 : \varphi)\} \subseteq \beta_2$.

□

Apresentamos agora novas definições de confutação e de confirmação para tableaux onde se terão em conta apenas os ramos finitos e os ramos com um ciclo.

Proposição 4.4.13 (Confutação e Confirmação de Tableaux)
Um tableaux $\mathcal{T}$ com raiz Ψ diz-se uma confirmação sse contém um ramo finito aberto ou um ramo com um ciclo aberto. O tableau diz-se uma confutação se apenas contém ramos finitos fechados, ramos com ciclos fechados e ramos com ciclos para o passado.

Demonstração:
Se todos os ramos do tableau são finitos e não contêm um ciclo, então a proposição verifica-se trivialmente. Vamos considerar que o tableau contém um ramo β_c com um ciclo para o futuro caracterizado pelos nós η_1, η_2 e η_3. Considerando os dois tipos de ciclo para o futuro,

- Se o ciclo é aberto, então é possível encontrar um ramo infinito aberto. Esse ramo irá ser obtido em seguida, supondo que ν_1, ν_2, ν_3 são os limites superiores dos nós anteriores e que $(\nu_1 \alpha^p \nu_2) \in \beta_c$ para um natural p.

  Seja $\{\mathcal{B}_n\}_{n \in I\!N_o}$ uma família de subconjuntos de $ramos(\mathcal{T})$ com a seguinte definição

  $$\mathcal{B}_0 = \{\beta \in ramos(\mathcal{T}) : \ \eta_1 \in \beta\},$$

  $$\mathcal{B}_{i+1} = \Big\{\beta \in \mathcal{B}_i : \ \{(\nu_1 \alpha^{resto(i+1,p)} \omega), (\omega : \varphi)\} \subseteq \beta \text{ sse}$$

  $$\{(\nu_1 \alpha^{i+1} \omega'), (\omega' : \varphi)\} \subseteq \beta_c\Big\},$$

  onde $resto(i,p)$ devolve o resto da divisão inteira de i por p.

  De um modo semelhante à demonstração da proposição anterior, verifica-se que os conjuntos da família $\{\mathcal{B}_n\}_{n \in I\!N_o}$ não são vazios. Ao fazer a intersecção dos conjuntos,

  $$\mathcal{B} = \bigcap_{n \in I\!N_o} \mathcal{B}_n,$$

  obtemos um conjunto não vazio de ramos do tableau e esse conjunto contém um ramo aberto. Podemos então concluir que o tableau é uma confirmação.

- Se o ciclo é fechado, podemos ainda considerar dois casos:
  - O ramo que o contém é fechado; neste caso, se não existir nenhum outro ramo aberto, o tableau é uma confutação.
  - O ramo que o contém é aberto; então existe um prefixo $\omega_1 \in \Omega$ e um natural $n \in I\!N_o$ tal que $\{(\nu_c \alpha^n \omega_1), (\omega_1 : \varphi_2)\} \subset \beta$. Recorrendo à proposição 4.4.12, existe um ramo aberto β' distinto de β e um prefixo $\omega_2 \in \Omega$ tal que $\{(\nu_a \alpha^n \omega_2), (\omega_2 : \varphi_2)\} \subset \beta'$. Consequentemente, o tableau é uma confirmação.

O caso em que ramo contém um ciclo para o passado é semelhante ao caso em que o ramo contém um ciclo para o futuro, fechado.

□

Para finalizar este capítulo, apresentamos um algoritmo para determinar se um tableau é uma confutação ou uma confirmação.

Observação 4.4.14 (Algoritmo de Decisão)
Sejam $\Gamma = \langle \Sigma, \Omega, \alpha \rangle$ uma assinatura de prefixos e Ψ um subconjunto finito de F_Γ tal que $pref(\Psi) = \{\omega\}$. O algoritmo começa com uma árvore que contém somente a raiz etiquetada com $\Psi \cup \Upsilon^\Sigma$. Os passos abaixo são aplicados apenas a ramos que não contenham um absurdo ou um ciclo. Seja $\mathcal{T}_0 = \langle Avr_0, et_0 \rangle$ tal que $Arv_0 = \{\varepsilon\}$ e $et_0(\varepsilon) = \Psi \cup \Upsilon^\Sigma_\omega$, o valor inicial de i é 0.

1. escolher um ramo β de $\mathcal{T}_i$ que não contenha um absurdo nem um ciclo e que não se encontre fechado para todas as regras de inferência de RI_Γ. Caso não exista um ramo que verifique estas condições, terminar o algoritmo;

2. se β não se encontra fechado para uma regra ϱ unária de RI_Σ diferente de $\mathsf{X}\,i$ e de $\mathsf{Y}\,i$, repetir o passo 1 considerando o tableau $\mathcal{T}_{i+1}$ definido do seguinte modo:

- $Arv_{i+1} = Arv_i \cup \{\eta_\beta.1\}$,
- $et_{i+1}(\eta) = \begin{cases} perm(\varrho) & \text{se } \eta = \eta_\beta.1 \\ et_i(\eta) & \text{se } \eta \in Avr_i \end{cases}$

3. se β não se encontra fechado para uma regra ϱ n-ária de RI_Σ $(n > 1)$, repetir o passo 1 considerando o tableau $\mathcal{T}_{i+1}$ definido do seguinte modo:

   - $Arv_{i+1} = Arv_i \cup \{\eta_\beta.1, \ldots, \eta_\beta.n\}$,
   - $et_{i+1}(\eta) = \begin{cases} perm_i(\varrho) & \text{se } \eta = \eta_\beta.i \\ et_i(\eta) & \text{se } \eta \in Avr_i \end{cases}$

4. seja ϱ uma regra de inferência de RI_Σ do tipo $\mathsf{Y}\, i$ com conclusão $et_i(\eta_\beta)$; repetir o passo 1 considerando o tableau $\mathcal{T}_{i+1}$ definido do seguinte modo:

   - $Arv_{i+1} = Arv_i \cup \{\eta_\beta.1\}$,
   - $et_{i+1}(\eta) = \begin{cases} perm(\varrho) & \text{se } \eta = \eta_\beta.1 \\ et_i(\eta) & \text{se } \eta \in Avr_i \end{cases}$

5. seja ϱ uma regra de inferência de RI_Γ do tipo $\mathsf{X}\, i$ com conclusão $et_i(\eta_\beta)$; repetir o passo 1 considerando o tableau $\mathcal{T}_{i+1}$ definido do seguinte modo:

   - $Arv_{i+1} = Arv_i \cup \{\eta_\beta.1\}$,
   - $et_{i+1}(\eta) = \begin{cases} perm(\varrho) & \text{se } \eta = \eta_\beta.1 \\ et_i(\eta) & \text{se } \eta \in Avr_i \end{cases}$

$\square$

O objectivo do algoritmo anterior é obter o mais rapidamente possível ramos fechados no tableau. As primeiras regras a serem aplicadas são as regras unárias à excepção das regras $\mathsf{X}\, i$ e $\mathsf{Y}\, i$. Esta opção tem por objectivo evitar bifurcações desnecessárias do tableau e obter as condições necessárias para definir nós limitados inferiormente e superiormente. As regras não unárias só são aplicadas caso não seja possível aplicar nenhuma das regras que verificam as condições anteriores. As regras $\mathsf{X}\, i$ e $\mathsf{Y}\, i$ são aplicadas em ramos que contêm nós limitados, mas que não contêm um ciclo. Entre estas duas regras é preferível aplicar primeiro a segunda, uma vez que os ciclos para o passado são mais fáceis de caracterizar e são todos fechados. Por fim é aplicada a regra $\mathsf{X}\, i$.

Uma vez que a raiz do tableau contém um número finito de fórmulas, o tableau em si também contém um número finito de fórmulas. Como tal, não existe nenhum ramo infinito que não contenha um ciclo. Sendo assim, o algoritmo acima termina. Seja $\mathcal{T}_n$ o último tableau construído pelo algoritmo; se esse tableau contém um ramo aberto ou um ramo com um ciclo aberto, então o tableau é um confirmação. Caso contrário, o tableau é um confutação.

Apresentamos em seguida um exemplo de um tableau construído de acordo com o algoritmo anterior. Optamos por definir uma nova assinatura de objecto, uma vez que as assinaturas apresentadas no capítulo anterior originariam tableaux demasiado extensos.

Exemplo 4.4.15 (Flip Flop)
Consideramos a especificação de objecto $\texttt{Flip\_Flop} = \langle \Sigma, Ax\rangle$ com a seguinte definição:

- $\Sigma = \langle \mathbb{Q}_o^+, 2, GACT, GATT\rangle$ em que:
  - $GACT = \langle ACT, par, dur\rangle$,
    - $ACT = \{\texttt{flip}, \texttt{flop}, \perp\}$,
    - $par(\texttt{flip}) = par(\texttt{flop}) = \emptyset$,
    - $dur(\texttt{flip}) = dur(\texttt{flop}) = 0$,
  - $GACT = \langle ATT, res, ANM\rangle$,
    - $ATT = \{\texttt{st}, occ_i, occ_f\}$,
    - $res(\texttt{st}) = \mathcal{N}_1$,
    - $res(occ_i) = res(occ_f) = \kappa$,
    - $ANM = \{(occ_i, occ_f)\}$,
- Ax contém as seguintes fórmulas:
  1. $(\mathsf{G}((occ_i = \texttt{flip}) \Rightarrow (\texttt{st} = 0)))$,
  2. $(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\texttt{st} = 1)))$,
  3. $(\mathsf{G}((occ_i = \texttt{flip}) \Rightarrow (\mathsf{X}(\texttt{st} = 1))))$,
  4. $(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\texttt{st} = 0))))$,
  5. $(\mathsf{G}((occ_i = \texttt{flip}) \Rightarrow (\mathsf{F}_{\preceq 5}(occ_i = \texttt{flop}))))$,
  6. $(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip}))))$

$\square$

Verifiquemos que

$$\texttt{Flip\_Flop} \vdash_\Sigma (\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 4}(\texttt{st} = 1))))).$$

$(1)\ 1_{\omega_0}, \ldots, (6)\ 6_{\omega_0}$

$(7)\ (\omega_0 : (\mathsf{G}((occ_i = \mathtt{flip}) \Leftrightarrow (occ_f = \mathtt{flip}))))$

$(8)\ (\omega_0 : (\mathsf{G}((occ_i = \mathtt{flop}) \Leftrightarrow (occ_f = \mathtt{flop}))))$

$(9)\ (\omega_0 : (\neg(\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 4}(\mathtt{st} = 1)))))))$

$\mathsf{G}\ (6)$

$(10)\ (\omega_0 : ((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip}))))$

$(11)\ (\omega_0 : (\neg(\mathsf{X}(\neg(\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip}))))))))$

$\neg\mathsf{G}\ (9)$

$(12)\ (\omega_0 : (\neg((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 4}(\mathtt{st} = 1))))))$

$\neg\mathsf{G}\ (9)$

$(13)\ (\omega_0 : ((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 4}(\mathtt{st} = 1)))))$

$(14)\ (\omega_0 : (\mathsf{X}(\neg(\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 4}(\mathtt{st} = 1)))))))$

$\neg\Rightarrow\ (12)$

$(15)\ (\omega_0 : (occ_i = \mathtt{flop}))$

$(16)\ (\omega_0 : (\neg(\mathsf{X}(\mathsf{F}_{\preceq 4}(\mathtt{st} = 1)))))$

$\Rightarrow\ (10)$

$(17)\ (\omega_0 : (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip})))$

$\Rightarrow\ (10)$

$(18)\ (\omega_0 : (\neg(occ_i = \mathtt{flop})))$

$\mathtt{X}\ (15)(18)$

$\mathsf{F}_{\preceq\delta}\, ii\ (17)$

$(19)\ (\omega_0 : (\neg(occ_i = \mathtt{flip})))$

$(20)\ (\omega_0 : (\mathsf{X}(\mathsf{F}_{\preceq 1}(occ_i = \mathtt{flip}))))$

$\mathsf{X}\, i\ (20)$

$(22)\ (\omega_0 \alpha \omega_1)$

$(23)\ (\omega_1 : (\mathsf{F}_{\preceq 1}(occ_i = \mathtt{flip})))$

$\mathsf{F}_{\preceq\delta}\, i\ (23)$

$(26)\ (\omega_1 : (occ_i = \mathtt{flip}))$

$\neg\mathsf{X}\ (11)$

$(27)\ (\omega_1 : (\neg(\neg(\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip})))))))$

$\neg\neg\ (27)$

$(28)\ (\omega_1 : (\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip})))))$

$\mathsf{F}_{\preceq\delta}\, ii\ (17)$

$(21)\ (\omega_0 : (occ_i = \mathtt{flip})$

$=iv\ (21)$

$(24)\ (\omega_0 : (\neg(occ_i = \mathtt{flop})))$

$(25)\ (\omega_0 : (\neg(occ_i = \bot)))$

$\mathtt{X}\ (15)(24)$

$$(28)\ (\omega_1 : (\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip})))))$$

G (28)

$$(29)\ (\omega_1 : ((occ_i = \texttt{flip}) \Rightarrow (\mathsf{X}(\texttt{st} = 1))))$$
$$(30)\ (\omega_1 : (\neg(\mathsf{X}(\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip}))))))))$$

$\neg\mathsf{X}$ (16)

$$(31)\ (\omega_1 : (\neg(\mathsf{F}_{\preceq 4}(\texttt{st} = 1))))$$

$\neg\,\mathsf{F}_{\preceq\delta}\,ii$ (31)

$$(32)\ (\omega_1 : (\neg(\texttt{st} = 1)))$$
$$(33)\ (\omega_1 : (\neg(\mathsf{X}(\mathsf{F}_{\preceq 2}(\texttt{st} = 1))))))$$

$\Rightarrow$ (29) | $\Rightarrow$ (29)

$$(34)\ (\omega_1 : (\mathsf{X}(\texttt{st} = 1)))$$

$$(35)\ (\omega_1 : (\neg(occ_i = \texttt{flip})))$$
X (26)(35)

$\mathsf{X}\,i$ (34)

$$(36)\ (\omega_1 \alpha \omega_2)$$
$$(37)\ (\omega_2 : (\texttt{st} = 1))$$

$\neg\mathsf{X}$ (33)

$$(38)\ (\omega_2 : (\neg(\mathsf{F}_{\preceq 2}(\texttt{st} = 1))))$$

$\neg\,\mathsf{F}_{\preceq\delta}\,ii$ (38)

$$(39)\ (\omega_2 : (\neg(\texttt{st} = 1)))$$
$$(40)\ (\omega_2 : (\neg(\mathsf{X}(\mathsf{F}_{\preceq 0}(\texttt{st} = 1)))))$$
X(37)(39)

Os ramos finitos que contêm um absurdo são indicados com uma cruz seguida das fórmulas que caracterizam o absurdo.

Para verificar a consequência sintáctica é apenas necessário considerar algumas das fórmulas da raiz. Para evitar a construção de um tableau extenso, ignoramos as restantes fórmulas da raiz. Assim, podemos considerar que o nó que contém as fórmulas (13) e (14) é um nó limitado superiormente pelo prefixo ω_0. Em seguida apresentamos a continuação do ramo que ficou incompleto.

$$(13)\ (\omega_0 : ((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1)))))$$
$$(14)\ (\omega_0 : (\mathsf{X}(\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1))))))))$$

$\mathsf{X}\, i$ (14)

$$(41)\ (\omega_0 \alpha \omega_1)$$
$$(42)\ (\omega_1 : (\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1)))))))$$

$\neg\mathsf{X}$ (11)

$$(43)\ (\omega_1 : (\neg(\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip})))))))$$

$\neg\neg$ (43)

$$(44)\ (\omega_1 : (\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip})))))$$

G (44)

$$(45)\ (\omega_1 : ((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip}))))$$
$$(46)\ (\omega_1 : (\neg(\mathsf{X}(\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip}))))))))$$

Left branch: $\neg\mathsf{G}$ (42)

$$(47)\ (\omega_1 : ((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1)))))$$
$$(48)\ (\omega_1 : (\mathsf{X}(\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1))))))))$$

$\mathsf{X}\, i$ (48)

$$(50)\ (\omega_1 \alpha \omega_2)$$
$$(51)\ (\omega_2 : (\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1)))))))$$

$\neg\mathsf{X}$ (46)

$$(52)\ (\omega_2 : (\neg(\neg(\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip})))))))$$

$\neg\neg$ (52)

$$(53)\ (\omega_2 : (\mathsf{G}((occ_i = \texttt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \texttt{flip})))))$$

Right branch: $\neg\mathsf{G}$ (42)

$$(49)\ (\omega_1 : (\neg((occ_i = \texttt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\texttt{st} = 1))))))$$

$\vdots$

X

$$(53)\ (\omega_2 : (\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip})))))$$

$$\Big|\ \mathsf{G}\ (53)$$

$$(54)\ (\omega_2 : ((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip}))))$$

$$(55)\ (\omega_2 : (\neg(\mathsf{X}(\neg(\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{F}_{\preceq 3}(occ_i = \mathtt{flip}))))))$$

$$\Big|\ \neg\,\mathsf{G}\ (50) \qquad\qquad \diagdown\ \neg\,\mathsf{G}\ (50)$$

$$(56)\ (\omega_2 : ((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\mathtt{st} = 1)))))$$

$$(57)\ (\omega_2 : (\mathsf{X}(\neg(\mathsf{G}((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\mathtt{st} = 1))))))))$$

$$(58)\ (\omega_2 : (\neg((occ_i = \mathtt{flop}) \Rightarrow (\mathsf{X}(\mathsf{F}_{\preceq 10}(\mathtt{st} = 1))))))$$

$$\vdots$$

$$\mathtt{X}$$

Os dois ramos à direita apresentavam fórmulas semelhantes às de ramos anteriores a menos de uma mudança de prefixos. Neste caso, todos os ramos por eles obtidos irão ser ramos fechados. Para evitar uma sobrecarga do tableau, optámos por classificar esses ramos como contendo um absurdo.

Consideremos agora o ramo β central. Ao analisar esse ramo, verifica-se que existem dois prefixos ω_1 e ω_2 que limitam superiormente dois nós. Temos portanto que

- $form(\beta, \omega_1) = form(\beta, \omega_2)$,
- $(\mathcal{V} \circ form_{\mathcal{F}})(\beta, \omega_2) = 0$,
- $(\omega_1 \alpha \omega_2), (\omega_2 \alpha^0 \omega_2) \in \beta$.

As propriedades de um ciclo para o futuro são satisfeitas. Podemos verificar que esse ciclo é fechado, o que permite concluir que o tableau é um confutação e que a consequência sintáctica é verdadeira.

Capítulo 5

Agregação de Objectos

5.1 Introdução

Neste capítulo faremos um estudo categorial sobre a lógica apresentada nos capítulos anteriores. Serão definidas a categoria das assinaturas de objectos e a categoria das especificações de objectos. Este estudo permite efectuar a agregação de especificações de objectos e raciocionar sobre essa agregação. Iremos seguir de perto o estudo apresentado em [Sernadas e Sernadas 94], efectuando as devidas alterações.

Antes de definir as categorias, apresentamos um resultado essencial para a continuação deste capítulo. Ao definir uma categoria entre as assinaturas, é necessário definir também os morfismos da categoria. Os morfismos traduzem acções e atributos de uma assinatura para outra, respeitando certas propriedades que serão apresentadas mais adiante. A tradução de acções irá ter influência sobre alguns elementos do domínio de valores $\mathcal{D}_{ob}$, uma vez que $\mathcal{D}_{ob}$ contém conjuntos formados por instanciações de acções. Fixada uma aplicação entre as acções de uma assinatura Σ_1 e as acções de uma assinatura Σ_2, é possível definir uma aplicação que traduza elementos do domínio de valores que contém acções da assinatura Σ_1 para elementos do domínio de valores que contém acções da assinatura Σ_2.

Definição 5.1.1 (Tradução de Domínios)
Sejam ACT_1, ACT_2 os conjuntos das acções das assinaturas de objectos Σ_1, Σ_2, respectivamente, e $f : ACT_1 \to ACT_2$ uma aplicação. A aplicação $f^{\mathcal{D}} : \mathcal{D}_1 \to \mathcal{D}_2$ é definida do seguinte modo:

$$f^{\mathcal{D}}(D) = \{\hat{f}(d) : \; d \in D\}$$

onde

$$\hat{f}(d) = \begin{cases} d & \text{se } d \notin \kappa_1 \\ f(act) & \text{se } d = act \text{ para } act \in ACT_1 \\ f(act)(d') & \text{se } d = act(d') \text{ para } act \in ACT_1 \end{cases}$$

□

Se um domínio contém uma acção da assinatura Σ_1, ao passarmos para a assinatura Σ_2 é conveniente traduzir as acções da primeira assinatura para acções da segunda. Essa tradução tem de respeitar a função f.

É possível estabelecer uma propriedade sobre a função de tradução. Para estabelecer esta propriedade é conveniente ter em consideração que $D \cap \kappa_1 = \emptyset$ e que $D \cap \kappa_2 = \emptyset$, para qualquer elemento $D \in \mathcal{D}$.

Proposição 5.1.2 Seja $f : ACT_1 \to ACT_2$ uma aplicação das acções da assinatura Σ_1 para as acções da assinaturas Σ_2. Se a aplicação f é injectiva, então a aplicação $f^{\mathcal{D}}$ também é injectiva.

Demonstração:
Sendo f injectiva, verifica-se com facilidade que $\hat{f}$ também é injectiva. Se $f^{\mathcal{D}}(D_1) = f^{\mathcal{D}}(D_2) = D$, então, para todo o $d_1 \in D_1$, existe um $d_2 \in D_2$ tal que $\hat{f}(d_1) = \hat{f}(d_2)$ e, para qualquer $d_2' \in D_2$, existe um $d_1' \in D_1$ tal que $\hat{f}(d_2') = \hat{f}(d_1')$. Pela injectividade de $\hat{f}$, temos que $d_1 = d_2$ e $d_1' = d_2'$, logo $D_1 = D_2$.

□

Definimos uma aplicação que efectua a passagem entre os elementos do domínio de valores para a assinatura Σ_1, para os elementos do domínio de valores para a assinatura Σ_2, respeitando uma aplicação, previamente estabelecida, entre os conjuntos das acções. Se a aplicação entre acções é injectiva, então a tradução de domínios também é injectiva.

5.2 Categoria de Especificação de Objectos

Nesta secção introduzimos a categoria das assinaturas de objectos e das especificações de objectos. Os conceitos utilizados foram retirados de [Adámek *et al* 90], podendo ser encontrados no apêndice A. Ao longo desta secção faremos referências a determinadas categorias que se encontram definidas na secção A.1.2.

Para definir a categoria das assinaturas é necessário caracterizar os seus morfismos.

Definição 5.2.1 (Morfismos entre Assinaturas)
Sejam $\Sigma_1 = \langle \Delta_1, \tau_1, GACT_1, GATT_1 \rangle$ e $\Sigma_2 = \langle \Delta_2, \tau_2, GACT_2, GATT_2 \rangle$ duas assinaturas de objectos; $\sigma : \Sigma_1 \to \Sigma_2$ diz-se um morfismo entre as assinaturas Σ_1 e Σ_2 se

- σ é um par $\langle \sigma_{ACT}, \sigma_{ATT} \rangle$ de aplicações onde:
  - $\sigma_{ACT} : ACT_1 \to ACT_2$ é uma aplicação entre acções tal que $\sigma_{ACT}(\perp) = \perp$,
  - $\sigma_{ATT} : ATT_1 \to ATT_2$ é uma aplicação entre atributos,
- $\Delta_1 \subseteq \Delta_2$,
- $\tau_2 |_\iota \tau_1$,
- $par_2 \circ \sigma_{ACT} = par_1$,
- $mul(\tau_1, dur_1(act)) \leq_\iota dur_2 \circ \sigma_{ACT}(act) <_\iota mul(\tau_1, dur_1(act)) +_\iota \tau_2$
  para qualquer $act \in ACT_1$,
- $res_2 \circ \sigma_{ATT} = \sigma^{\mathcal{D}}_{ACT} \circ res_1$,
- se $(att_1, att_2) \in ANM_1$, então $(\sigma_{ATT}(att_1), \sigma_{ATT}(att_2)) \in ANM_2$.

□

Existe um morfismo entre duas assinaturas de objectos se a segunda assinatura contém toda a informação da primeira. Deste modo, um morfismo entre duas assinaturas de objectos é constituído por um par de aplicações: uma entre as acções de cada assinatura e outra entre os respectivos atributos. Estas aplicações têm de respeitar os parâmetros das acções, os resultados dos atributos, os tempos de duração das acções e também os pares de atributos responsáveis pela animação do objecto. Em relação à duração das acções não é exigida a igualdade na aplicação *dur* entre as duas assinaturas. Tendo em conta que o tempo se encontra repartido por intervalos com comprimento igual ao período das assinaturas, a desigualdade apresentada é suficiente para garantir o preservação da duração de cada acção. No que respeita aos atributos é necessário efectuar a tradução dos resultados dos atributos da primeira assinatura para a segunda, respeitando a aplicação σ_{ACT} entre acções.

Ao nível do tempo, é necessário que o domínio de tempo da assinatura Σ_1 esteja contido no domínio do tempo da assinatura Σ_2 e que o período do relógio da primeira assinatura seja um múltiplo inteiro do período do relógio da segunda. Esta última condição permite garantir que os sucessivos períodos do primeiro relógio coincidem com os do segundo, apesar de este apresentar um período menor.

No seguimento, designaremos por $\sigma^{\mathcal{D}}$ a aplicação $\sigma^{\mathcal{D}}_{ACT}$.

Uma vez caracterizados os morfismos entre assinaturas, iremos definir a categoria das assinaturas de objectos. Esta categoria tem a propriedade de ser co-cartesiana.

Prop/Definição 5.2.2 (Categoria das Assinaturas de Objectos)
As assinaturas de objectos e os morfismos entre assinaturas formam a categoria co-cartesiana $\mathcal{A}sn$ das assinaturas.

Demonstração:
Verifiquemos que a categoria contém objecto inicial e co-produtos binários:

- objecto inicial: $1_{Asn} = \langle \emptyset, \iota, GACT, GATT \rangle$ onde :
  - $GACT = \langle \{\perp\}, par, dur \rangle$ em que $par(\perp) = \emptyset$ e $dur(\perp) = 0$,
  - $GATT = \langle \emptyset, res, \emptyset \rangle$.
- o co-produto de $\Sigma_1 = \langle \Delta_1, \tau_1, GACT_1, GATT_1 \rangle$ e de $\Sigma_2 = \langle \Delta_2, \tau_2, GACT_2, GATT_2 \rangle$ é definido do seguinte modo:

$$\Sigma_1 \oplus \Sigma_2 = \langle \Sigma, \imath_1, \imath_2 \rangle$$

onde

- $\langle \langle ACT, \perp \rangle, \imath_{1ACT}, \imath_{2ACT} \rangle$ é o co-producto de $\langle ACT_1, \perp \rangle$ e de $\langle ACT_2, \perp \rangle$ na categoria $\mathcal{S}et\mathcal{P}nt$,
- $\langle ATT, \imath_{1ATT}, \imath_{2ATT} \rangle$ é o co-produto de ATT_1 e ATT_2 na categoria $\mathcal{S}et$,
- $\imath_1 = \langle \imath_{1ACT}, \imath_{1ATT} \rangle$,
- $\imath_2 = \langle \imath_{2ACT}, \imath_{2ATT} \rangle$,
- $par : ACT \rightarrow \mathcal{D}$ é tal que $par \circ \imath_{iACT} = par_i$ para $i = 1, 2$,
- $dur : ACT \rightarrow \Delta \cup \{0\}$ é tal que
  $$dur \circ \imath_{iACT}(act) = mul(\tau_i, dur_i(act))$$
  para toda a acção $act \in ACT_i$ e $i = 1, 2$,
- $res : ATT \rightarrow \mathcal{D}_{ACT}$ é tal que $res \circ \imath_{iATT} = \imath_i^{\mathcal{D}} \circ res_i$ para $i = 1, 2$,
- $ANM = \{(\imath_{1ATT}(att_1), \imath_{1ATT}(att_2)) : (att_1, att_2) \in ANM_1\} \cup \{(\imath_{2ATT}(att_1), \imath_{2ATT}(att_2)) : (att_1, att_2) \in ANM_2\}$,
- $GACT = \langle ACT, par, dur \rangle$,
- $GATT = \langle ATT, res, ANM \rangle$,
- $\Sigma = \langle \Delta_1 \cup \Delta_2, mdc_g(\tau_1, \tau_2), GACT, GATT \rangle$.

□

O elemento inicial da categoria consiste na assinatura do objecto mais simples que existe. Essa assinatura só contém a acção $\perp$ com as respectivas aplicações par e dur. O cojunto dos atributos e dos pares de animação é vazio.

O co-produto de duas assinaturas é definido com base no co-produto em $\mathcal{S}et\mathcal{P}nt$ para as acções e com base no co-produto $\mathcal{S}et$ para os atributos. Estas categorias encontram-se definidas em A.1.2. Os diversos elementos da assinatura obtidos pelo co-produto encontram-se bem definidos, tendo em conta a definição de co-produto e que $\imath_i^{\mathcal{D}}$ é injectivo para $i = 1, 2$.

Com base na definição de morfismos entre assinaturas, podemos definir uma família de aplicações entre os termos das assinaturas.

Definição 5.2.3 (Aplicação de Tradução para Termos)
Seja $\sigma = \langle \sigma_{ACT}, \sigma_{ATT} \rangle$ um morfismo entre as assinaturas Σ_1 e Σ_2. A família das aplicações de tradução para termos induzida por σ, $\sigma_T = \{\sigma_{T,D} : T_{\Sigma_1,D} \rightarrow T_{\Sigma_2,\sigma^{\mathcal{D}}(D)}\}_{D \in \mathcal{D}_1}$, é constituída por aplicações definidas do seguinte modo:

- $\sigma_{T,D}(d) = d$ se $d \in D, D \in \mathcal{D}$ e $D \cap \kappa = \emptyset$,
- $\sigma_{T,D}(x) = x$ se $x \in X_D$,
- $\sigma_{T,res_1(att)}(att) = \sigma_{ATT}(att)$ se $att \in ATT_1$,
- $\sigma_{T,D}(act) = \sigma_{ACT}(act)$ se $act \in ACT_1$ e $par(act) = \emptyset$,
- $\sigma_{T,D}(act(t)) = \sigma_{ACT}(act)(\sigma_{T,par_1(act)}(t))$ se $act \in ACT_1$ e $t \in T_{\Sigma_1,par(act)}$.

$\square$

Uma aplicação entre os termos de duas assinaturas traduz atributos e acções de uma assinatura para a outra assinatura, respeitando o morfismo subjacente e mantendo constantes os restantes termos.

Para as fórmulas é possível definir uma aplicação que traduza fórmulas de uma assinatura para outra. Esta aplicação é mais complexa dos que a aplicação de tradução de termos, uma vez que irão existir fórmulas relacionadas com o tempo e, consequentemente, será necessário ter em consideração os períodos das duas assinaturas.

Definição 5.2.4 (Aplicação de Tradução para Fórmulas)
Seja $\sigma = \langle \sigma_{ACT}, \sigma_{ATT} \rangle$ um morfismo entre as assinaturas Σ_1 e Σ_2, com períodos τ_1 e τ_2, respectivamente. A aplicação de tradução para fórmulas induzida por σ, $\sigma_L : F_{\Sigma_1} \rightarrow F_{\Sigma_2}$, consiste numa aplicação com a seguinte definição:

- $\sigma_L(t = t') = (\sigma_{T,D}(t) = \sigma_{T,D}(t'))$ tal que $t, t' \in T_{\Sigma,D}$,
- $\sigma_L(\neg\, \varphi) = (\neg\, \sigma_L(\varphi))$,
- $\sigma_L(\varphi_2 \Rightarrow \varphi_2) = (\sigma_L(\varphi_1) \Rightarrow \sigma_L(\varphi_2))$,
- $\sigma_L(\forall x\ \varphi) = (\forall x\ \sigma_L(\varphi))$,
- $\sigma_L(\mathsf{X}\, \varphi) = \begin{cases} (\mathsf{X}\, \sigma_L(\varphi)) & \text{se } \tau_1 = \iota \text{ ou } \tau_1 = \tau_2, \\ (\mathsf{F}_{\tau_1}\, \sigma_L(\varphi)) & \text{se } \tau_1 \neq \iota \text{ e } \tau_1 \neq \tau_2 \end{cases}$
- $\sigma_L(\varphi_2\, \mathsf{U}_\delta\, \varphi_2) = \begin{cases} (\sigma_L(\varphi_1)\, \mathsf{U}_\delta\, \sigma_L(\varphi_2)) & \text{se } \tau_1 = \tau_2, \\ (\sigma_L(\varphi_1)\, \mathsf{U}_{mul(\tau_1,\delta)}\, \sigma_L(\varphi_2)) & \text{se } \tau_1 \neq \tau_2 \end{cases}$

- $\sigma_L(\varphi_2 \, \mathsf{U}_{\preceq\delta} \, \varphi_2) = \begin{cases} (\sigma_L(\varphi_1) \, \mathsf{U}_{\preceq\delta} \, \sigma_L(\varphi_2)) & \text{se } \tau_1 = \tau_2, \\ (\sigma_L(\varphi_1) \, \mathsf{U}_{\preceq mul(\tau_1,\delta)} \, \sigma_L(\varphi_2)) & \text{se } \tau_1 \neq \tau_2 \end{cases}$

- $\sigma_L(\mathsf{Y} \, \varphi) = \begin{cases} (\mathsf{Y} \, \sigma_L(\varphi)) & \text{se } \tau_1 = \iota \text{ ou } \tau_1 = \tau_2, \\ (\mathsf{P}_{\tau_1} \, \sigma_L(\varphi)) & \text{se } \tau_1 \neq \iota \text{ e } \tau_1 \neq \tau_2 \end{cases}$

- $\sigma_L(\varphi_2 \, \mathsf{S}_{\delta} \, \varphi_2) = \begin{cases} (\sigma_L(\varphi_2) \, \mathsf{S}_{\delta} \, \sigma_L(\varphi_2)) & \text{se } \tau_1 = \tau_2 \\ (\sigma_L(\varphi_1) \, \mathsf{S}_{mul(\tau_1,\delta)} \, \sigma_L(\varphi_2)) & \text{se } \tau_1 \neq \tau_2 \end{cases}$

- $\sigma_L(\varphi_2 \, \mathsf{S}_{\preceq\delta} \, \varphi_2) = \begin{cases} (\sigma_L(\varphi_2) \, \mathsf{S}_{\preceq\delta} \, \sigma_L(\varphi_2)) & \text{se } \tau_1 = \tau_2 \\ (\sigma_L(\varphi_1) \, \mathsf{S}_{\preceq mul(\tau_1,\delta)} \, \sigma_L(\varphi_2)) & \text{se } \tau_1 \neq \tau_2 \end{cases}$

□

Para as fórmulas com conectivos lógicos, a sua imagem é a imagem da(s) subfórmula(s) enquadrada(s) no conectivo. No caso dos operadores temporais, é necessário relacionar os períodos das duas assinaturas.

Consideremos a fórmula $(\mathsf{X} \, \varphi)$; diz-se que esta fórmula se verifica no estado w se existe um estado w' que seja consecutivo a w e se φ se verifica em w'. Ao nível do tempo, a distância entre os estados anteriores é τ_1. Se os períodos das assinaturas coincidirem, então dizemos que dois estados consecutivos se encontram à mesma distância em tempo e, portanto, que a imagem da fórmula $(\mathsf{X} \, \varphi)$ será a imagem da sua subfórmula enquadrada pelo operador temporal X. Caso os períodos não coincidam, então dois estados consecutivos da primeira assinatura não se encontram à mesma distância em tempo dos dois estados consecutivos da segunda assinatura. Para respeitar a distância no tempo imposta pelo operador X, é necessário indicar explicitamente que a fórmula φ é verificada num estado a distância τ_1.

Para a fórmula $(\varphi_1 \, \mathsf{U}_{\delta} \, \varphi_2)$, o raciocínio é semelhante. Neste caso, e considerando períodos distintos, o estado onde a fórmula φ_2 é verificada terá de estar a uma distância em tempo múltipla de τ_1 e o mais próximo possível de δ. Essa distância é $mul(\tau_1, \delta)$.

Para os operadores temporais para o passado é efectuado o mesmo raciocínio na tradução das suas fórmulas.

Definida a categoria das assinaturas de objectos, iremos agora definir a categoria das especificações de objectos. Mas primeiro apresentamos a definição de morfismo entre duas especificações. Esta definição irá recorrer à tradução de fórmulas.

Definição 5.2.5 (Morfismo entre Especificações de Objectos)
Sejam $Ob_1 = \langle \Sigma_1, Ax_1 \rangle$ e $Ob_2 = \langle \Sigma_2, Ax_2 \rangle$ duas especificações de objectos. Então $\sigma : Ob_1 \to Ob_2$ diz-se um morfismo entre as especificações Ob_1 e Ob_2 sse:

- $\sigma : \Sigma_1 \to \Sigma_2$ é um morfismo em $\mathcal{A}sn$,

- $Ob_2 \vDash_{\Sigma_2} \sigma_L(\varphi)$ para qualquer $\varphi \in Ax_1$.

$\square$

Diz-se que existe um morfismo entre duas especificações de objectos se a segunda verifica todos os axiomas da primeira. Isto é, se a especificação do segundo objecto simula a especificação do primeiro.

Apresentamos em seguida a categoria das especificações de objectos. Esta categoria, tal como a categoria das assinaturas, é co-cartesiana.

Prop/Definição 5.2.6 (Categoria de Especificações de Objectos)
As especificações de objectos para a assinatura Σ e os seus morfismos constituem uma categoria co-cartesiana das especificações de objectos, $\mathcal{O}bj$.

Demonstração:
Verifiquemos que a categoria contém objecto inicial e co-produtos binários:

- objecto inicial: $1_{\mathcal{O}bj} = \langle 1_{\mathcal{A}sn}, \emptyset \rangle$,
- o co-produto de $Ob_1 = \langle \Sigma_1, Ax_1 \rangle$ e de $Ob_2 = \langle \Sigma_2, Ax_2 \rangle$ é definido do modo seguinte,

$$Ob_1 \oplus Ob_2 = \langle \langle \Sigma, Ax \rangle, \imath_1, \imath_2, \rangle$$

onde

- $\langle \Sigma, \imath_1, \imath_2 \rangle$ é o co-produto de Σ_1 e Σ_2 em $\mathcal{A}sn$,
- $Ax = \{\imath_{1L}(\varphi) : \ \varphi \in Ax_1\} \cup \{\imath_{2L}(\varphi) : \ \varphi \in Ax_2\}$.

$\square$

A especificação de objecto inicial é constituída pela assinatura inicial da categoria $\mathcal{A}sn$, sendo o conjunto dos axiomas vazio. O co-produto entre duas especificações é formado pelo co-produto das assinaturas e pela união das imagens dos axiomas de cada especificação.

5.3 Comunidade de Objectos

Nesta secção será introduzida a noção de comunidade de objecto. A sintaxe e a semântica serão apresentadas, bem como um sistema dedutivo para raciocinar com uma comunidade.

5.3.1 Sintaxe e Semântica

Definição 5.3.1 (Assinatura de Comunidade)
Seja Pop um conjunto finito chamado população e $\mathcal{C}_{Pop}$ a categoria cartesiana livremente gerada a partir de (Pop, id). A assinatura de comunidade de Pop consiste num par $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|}\rangle$ onde:

- se $r \in Pop$, então Σ_r é uma assinatura de objecto primário,
- se $r = r_1\|\ldots\|r_n$, então $\langle \Sigma_r, \imath_1, \imath_2\rangle$ é um co-produto de $\Sigma_{r_1}, \ldots, \Sigma_{r_n}$ em $\mathcal{A}sn$.

□

Uma assinatura de comunidade é constituída por um conjunto finito intitulado *população* e por uma família de assinaturas de objectos. A população da comunidade constitui a base da comunidade. As assinaturas de objectos etiquetadas pelos elementos da população são assinaturas de objectos primários. As restantes assinaturas são co-produtos destas.

Exemplo 5.3.2 Retomando o exemplo 3.2.3 da página 17, consideremos a população formada por $\{\texttt{pacote}, \texttt{tapete}, \texttt{disp\_bin}\}$. A categoria cartesiana definida a partir da população terá os seguintes objectos: `pacote`, `tapete`, `disp_bin`, `pacote||tapete`, `pacote||disp_bin`, `tapete||disp_bin` e `pacote||tapete||disp_bin`. As assinaturas para cada objecto de $|\mathcal{S}_{Pop}|$ consistem no co-produto das assinaturas das partes. Por exemplo, no caso de `pacote||disp_bin`, a sua assinatura é formada pelos segiuntes elementos:

- $\Delta_{pacote\|disp\_bin} = \mathbb{Q}_o^+$;
- $\tau_{pacote\|disp\_bin} = 2$;
- para $GACT_{pacote\|disp\_bin} = \langle ACT_{pacote\|disp\_bin}, par_{pacote\|disp\_bin}, dur_{pacote\|disp\_bin}\rangle$ temos:
  - $ACT_{pacote\|disp\_bin} = \{\texttt{obs\_id}^{pacote}, \star^{pacote}, \texttt{entr\_pct}^{disp\_bin}, \texttt{saída\_pct}_1{}^{disp\_bin}, \texttt{saída\_pct}_2{}^{disp\_bin}, \texttt{md\_pat}^{disp\_bin}, \star^{disp\_bin}, \perp\}$,
  - $par_{pacote\|disp\_bin}(act) = \begin{cases} par_{pacote}(act_1) & \text{se } act = act_1^{pacote} \\ par_{disp\_bin}(act_2) & \text{se } act = act_2^{disp\_bin} \\ \emptyset & \text{se } act = \perp \end{cases}$
  - $dur_{pacote\|disp\_bin}(act) = 0$ para qualquer $act \in ACT_{pacote\|disp\_bin}$
- para $GATT_{pacote\|disp\_bin} = \langle ATT_{pacote\|disp\_bin}, res_{pacote\|disp\_bin}, ANM_{pacote\|disp\_bin}\rangle$ temos:

- $ATT_{pacote||disp\_bin} = \{\mathtt{id}^{pacote}, occ_i^{pacote}, occ_f^{pacote}, \mathtt{direc}^{dist\_bin}, occ_i^{dist\_bin}, occ_f^{dist\_bin}\}$,
- $res_{pacote||disp\_bin}(act) = \begin{cases} \imath_1^{\mathcal{D}} \circ res_{pacote}(att_1) & \text{se } att = att_1^{pacote} \\ \imath_2^{\mathcal{D}} \circ res_{disp\_bin}(att_2) & \text{se } att = att_2^{dist\_bin} \end{cases}$

  onde $\imath_1$ e $\imath_2$ são morfismos na categoria $\mathcal{S}et$ que definem o co-produto $\langle ATT_{pacote||disp\_bin}, \imath_1, \imath_2\rangle$ de ATT_{pacote} e ATT_{disp_bin},
- $ANM = \{(occ_i^{pacote}, occ_f^{pacote}), (occ_i^{disp\_bin}, occ_f^{disp\_bin})\}$.

Para o elemento tapete||dist_bin, a assinatura será a seguinte:

- $\Delta_{tapete||disp\_bin} = \mathbb{Q}_o^+$,
- $\tau_{tapete||disp\_bin} = 1$;
- para $GACT_{tapete||disp\_bin} = \langle ACT_{tapete||disp\_bin}, par_{tapete||disp\_bin}, dur_{tapete||disp\_bin}\rangle$ temos:
  - $ACT_{tapete||disp\_bin} = \{\mathtt{perc}^{tapete}, \star^{tapete}, \mathtt{entr\_pct}^{disp\_bin}, \mathtt{saída\_pct_1}^{disp\_bin}, \mathtt{saída\_pct_2}^{disp\_bin}, \mathtt{md\_pat}^{disp\_bin}, \star^{disp\_bin}, \perp\}$,
  - $par_{tapete||disp\_bin}(act) = \begin{cases} par_{tapete}(act_1) & \text{se } act = act_1^{tapete} \\ par_{disp\_bin}(act_2) & \text{se } act = act_2^{disp\_bin} \\ \emptyset & \text{se } act = \perp \end{cases}$
  - $dur_{tapete||disp\_bin}(act) = \begin{cases} mul(5, dur_{tapete}(act_1)) & \text{se } act = act_1^{tapete} \\ mul(2, dur_{disp\_bin}(act_2)) & \text{se } act = act_2^{disp\_bin} \\ 0 & \text{se } act = \perp \end{cases}$
- para $GATT_{tapete||disp\_bin} = \langle ATT_{tapete||disp\_bin}, res_{tapete||disp\_bin}, ANM_{tapete||disp\_bin}\rangle$ temos:
  - $ATT_{tapete||disp\_bin} = \{occ_i^{tapete}, occ_f^{tapete}, occ_i^{dist\_bin}, occ_f^{dist\_bin}\}$,
  - $res_{tapete||disp\_bin}(act) = \begin{cases} \imath_1^{\mathcal{D}} \circ res_{tapete}(att_1) & \text{se } att = att_1^{tapete} \\ \imath_2^{\mathcal{D}} \circ res_{disp\_bin}(att_2) & \text{se } att = att_2^{dist\_bin} \end{cases}$

    onde $\imath_1$ e $\imath_2$ são morfismos na categoria $\mathcal{S}et$ que definem o co-produto $\langle ATT_{tapete||disp\_bin}, \imath_1, \imath_2\rangle$ de ATT_{tapete} e ATT_{disp_bin},
  - $ANM = \{(occ_i^{tapete}, occ_f^{tapete}), (occ_i^{disp\_bin}, occ_f^{disp\_bin})\}$.

$\square$

Uma comunidade é o enriquecimento de uma assinatura de comunidade com uma família de conjuntos de fórmulas temporais das várias lógicas para as assinaturas de objectos definidas na comunidade. A cada assinatura de objecto da assinatura da comunidade associa-se um conjunto finito de fórmulas.

Definição 5.3.3 (Comunidade)
Seja Pop uma população. Uma comunidade Com é um par $\langle \Sigma_{com}, \{\Phi_r\}_{r \in |\mathcal{C}_{Pop}|}\rangle$ onde:

- $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|}\rangle$ é uma assinatura de comunidade;
- Φ_r é um subconjunto finito de F_{Σ_r} tal que, se $r \in Pop$, então $\langle \Sigma_r, \Phi_r\rangle$ é uma especificação de objecto primário.

$\square$

O conjunto das fórmulas Φ_r tem, nesta definição, um papel duplo, consoante r é um elemento da população ou não. No primeiro caso, Φ_r contém as fórmulas que são satisfeitas pelo comportamento do objecto com assinatura Σ_r. No segundo caso, Φ_r contém as fórmulas de interacção entre os objectos subjacentes. No caso de o conjunto Φ_r ser vazio para $r = r_1 \| \ldots \| r_n$, não existe nenhuma restrição a impor aos objectos, dizendo-se que o objecto com assinatura Σ_r consiste no "interleaving" dos seus componentes.

Exemplo 5.3.4 Se considerarmos $r = \texttt{pacote}\|\texttt{dist\_bin}$, temos que

$$\Phi_{pacote\|dist\_bin} = \{(\mathsf{G}((occ_i^{dist\_bin} = \texttt{entr\_pct}^{dist\_bin}(x)) \Rightarrow (occ_i^{pacote} = \texttt{obs\_id}^{pacote}(x)))),$$

$$(\mathsf{G}((occ_i^{dist\_bin} = \texttt{saída\_pct}^{dist\_bin}(x)) \Rightarrow (occ_i^{pacote} = \texttt{obs\_id}^{pacote}(x))))\}.$$

Para $r = \texttt{tapete}\|\texttt{dist\_bin}$, já temos que

$$\Phi_{tapete\|dist\_bin} = \{(\mathsf{G}((occ_i^{dist\_bin} = \texttt{entr\_pct}^{dist\_bin}(x)) \Leftrightarrow (occ_i^{tapete} = \texttt{perc}^{tapete}(x)))),$$

$$(\mathsf{G}((occ_i^{dist\_bin} = \texttt{saída\_pct}_1^{dist\_bin}(x)) \Leftrightarrow (occ_f^{tapete} = \texttt{perc}^{tapete}(x)))),$$

$$(\mathsf{G}((occ_i^{dist\_bin} = \texttt{saída\_pct}_2^{dist\_bin}(x)) \Leftrightarrow (occ_f^{tapete} = \texttt{perc}^{tapete}(x))))\}.$$

$\square$

Na comunidade de objectos encontram-se definidas implicitamente as especificações dos objectos que a compõem. Quando se pretende obter explicitamente a especificação de um objecto da comunidade, é necessário ter em consideração, não só as especificações de objectos primários da comunidade, mas também as fórmulas de interacção para o objecto em causa.

Definição 5.3.5 (Especificação de um Objecto da Comunidade)
Seja $Com = \langle \Sigma_{com}, \{\Phi_r\}_{r \in |\mathcal{C}_{Pop}|} \rangle$ uma comunidade onde $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|} \rangle$. A especificação de objecto associado ao elemento $r \in |\mathcal{C}_{Pop}|$ é definida do seguinte modo:

- se $r \in Pop$, então $Ob_r = \langle \Sigma_r, \Phi_r \rangle$,
- se $r = r_1 \| \ldots \| r_n$, então $Ob_r = \langle \Sigma_r, \Phi \cup \Phi_r \rangle$ onde $Ob_{r_1} \oplus \ldots \oplus Ob_{r_n} = \langle \langle \Sigma_r, \Phi \rangle, \imath_1, \imath_2 \rangle$ é o co-produto de $Ob_{r_1}, \ldots, Ob_{r_n}$ em $\mathcal{O}bj$.

□

Exemplo 5.3.6 Se considerarmos r = $\mathtt{tapete} \| \mathtt{dist\_bin}$, a especificação de objecto $\langle \Sigma_{tapete\|dist\_bin}, Ax_{tapete\|dist\_bin} \rangle$ é constituída pela assinatura definida em 5.3.2 e pelos seguintes axiomas:

1. $(\mathsf{G}((occ_i^{dist\_bin} = \star^{dist\_bin}) \Rightarrow (\mathsf{X}(\mathtt{direc}^{dist\_bin} = left))))$,
2. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{entr\_pct}^{dist\_bin}(x)) \Rightarrow$
   $(\mathsf{F}((occ_i^{dist\_bin} = \mathtt{saída\_pct_1}^{dist\_bin}(x)) \vee$
   $(occ_i^{dist\_bin} = \mathtt{saída\_pct_2}^{dist\_bin}(x))))))$,
3. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{saída\_pct}_1^{dist\_bin}(x)) \Rightarrow (\mathsf{P}(occ_i^{dist\_bin} = \mathtt{entr\_pct}^{dist\_bin}(x)))))$,
4. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{saída\_pct_2}^{dist\_bin}(x)) \Rightarrow (\mathsf{P}(occ_i^{dist\_bin} = \mathtt{entr\_pct}^{dist\_bin}(x)))))$,
5. $(\mathsf{G}(((occ_i^{dist\_bin} = \mathtt{saída\_pct_1}^{dist\_bin}(x)) \vee (occ_i^{dist\_bin} = \mathtt{saída\_pct_2}^{dist\_bin}(x))) \Rightarrow$
   $(\mathsf{X}(occ_i^{dist\_bin} = \mathtt{md\_pat}^{dist\_bin}))))$,
6. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{md\_pat}^{dist\_bin}) \Rightarrow$
   $(\mathsf{Y}((occ_i^{dist\_bin} = \mathtt{saída\_pct_1}^{dist\_bin}) \vee (occ_i^{dist\_bin} = \mathtt{saída\_pct_2}^{dist\_bin})))))$,
7. $(\mathsf{G}(((occ_f^{dist\_bin} = \mathtt{md\_pat}^{dist\_bin}) \wedge (\mathtt{direc}^{dist\_bin} = left)) \Rightarrow$
   $(\mathsf{X}(\mathtt{direc}^{dist\_bin} = right))))$,
8. $(\mathsf{G}(((occ_f^{dist\_bin} = \mathtt{md\_pat}^{dist\_bin}) \wedge (\mathtt{direc}^{dist\_bin} = right)) \Rightarrow$
   $(\mathsf{X}(\mathtt{direc}^{dist\_bin} = left))))$,
9. $(\mathsf{G}((\mathtt{direc}^{dist\_bin} = left) \Leftrightarrow (\neg(\mathtt{direc}^{dist\_bin} = right))))$,
10. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{entr\_pct}^{dist\_bin}(x)) \Leftrightarrow (occ_i^{tapete} = \mathtt{perc}^{tapete}(x))))$,
11. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{saída\_pct}_1^{dist\_bin}(x)) \Leftrightarrow (occ_f^{tapete} = \mathtt{perc}^{tapete}(x))))$,
12. $(\mathsf{G}((occ_i^{dist\_bin} = \mathtt{saída\_pct}_2^{dist\_bin}(x)) \Leftrightarrow (occ_f^{tapete} = \mathtt{perc}^{tapete}(x))))$.

□

Fica assim completa a sintaxe para definir uma comunidade de objectos. Em seguida apresentamos a sua semântica.

Como uma assinatura de comunidade é composta por assinaturas de objectos, a semântica escolhida para interpretar a comunidade encontra-se fortemente relacionada com a semântica para as assinaturas de objectos. Efectivamente, uma estrutura de interpretação para uma assinatura de comunidade consiste numa família de estruturas de interpretação para as assinaturas de objectos da comunidade.

Definição 5.3.7 (Estrutura de Interpretação)
Seja $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|}) \rangle$ uma assinatura de comunidade. Uma estrutura de interpretação para a assinatura Σ_{com} é uma família $M_{com} = \{M_r\}_{r \in |\mathcal{C}_{Pop}|}$ de estruturas de interpretação para as assinaturas Σ_r.

□

Definida a semântica para uma assinatura de comunidade, apresentamos em seguida as noções de satisfação semântica e de consequência semântica.

Definição 5.3.8 (Satisfação Semântica)
Sejam $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|} \rangle$ uma assinatura de comunidade, $r \in |\mathcal{C}_{Pop}|$ e φ uma fórmula temporal de Σ_r. Diz-se que φ é satisfeita pela estrutura de interpretação de comunidade M_{com}, escrevendo-se $M_{com} \Vdash_{\Sigma_{com}} \varphi$, sse $M_r \Vdash_{\Sigma_r} \varphi$.

□

Definição 5.3.9 (Consequência Semântica)
Sejam $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|}) \rangle$ uma assinatura de comunidade, $r \in |\mathcal{C}_{Pop}|$, Φ um subconjunto de F_{Σ_r} e φ uma fórmula de F_{Σ_r}. Diz-se que φ é consequência semântica de Φ, escrevendo-se $\Phi \vDash_{\Sigma_{com}} \varphi$, sse $\Phi \vDash_{\Sigma_r} \varphi$.

□

A consequência semântica pode ser particularizada para uma comunidade de objectos.

Definição 5.3.10 (Consequência Semântica de uma Comunidade)
Sejam $Com = \langle \Sigma_{com}, \{\Phi_r\}_{r \in |\mathcal{C}_{Pop}|} \rangle$ uma comunidade onde $\Sigma_{com} = \langle Pop, \{\Sigma_r\}_{r \in |\mathcal{C}_{Pop}|} \rangle$, $r \in |\mathcal{C}_{Pop}|$ e $\varphi \in F_{\Sigma_r}$. Diz-se que a fórmula φ é consequência semântica da comunidade Com, escrevendo-se $Com \vDash_{\Sigma_{com}} \varphi$, sse $Ob_r \vDash_{\Sigma_r} \varphi$.

□

A partir de uma comunidade de objectos, é possível verificar fórmulas que não poderiam ser verificadas se considerássemos os diversos objectos isoladamente. Em particular, a fórmula $((occ_i^{dist\_bin} = \texttt{entr\_pact}^{dist\_bin}(x)) \Rightarrow (\neg(\mathsf{F}_{\preceq 4}(occ_i^{dist\_bin} = \texttt{md\_pat}^{dist\_bin}))))$ verifica-se na comunidade de objectos apresentada como exemplo, embora não seja possível verificá-la considerando isoladamente as três especificações de objectos.

5.3.2 Sistema Dedutivo

Tal como para a semântica de uma assinatura de comunidade, o sistema dedutivo será constituído por famílias de sistemas dedutivos para cada assinatura de objectos da comunidade.

Definição 5.3.11 (Sistema Dedutivo de uma Comunidade)
Seja $Com = \langle \Sigma_{com}, \{\Phi_r\}_{r \in |\mathcal{C}_{Pop}} \rangle$ uma comunidade. O sistema dedutivo SD_{com} da comunidade Com consiste numa família $\{SD_r\}_{rin|\mathcal{C}_{Pop}}$ de sistemas dedutivos $SD_r = \langle Ax_{\Sigma_r}, RI_{\Sigma_r} \rangle$ para a assinatura de objecto Σ_r.

$\square$

Dado um sistema dedutivo, é possível definir uma consequência sintáctica.

Definição 5.3.12 (Consequência Sintáctica de uma Comunidade)
Uma fórmula $\varphi \in F_{\Sigma_r}$ é consequência sintáctica da comunidade Com, escrevendo-se $Com \vdash_{com} \varphi$, sse $Ob_r \vdash_{\Sigma_r} \varphi$.

$\square$

Capítulo 6

Conclusões

Apresentou-se uma lógica temporal que permite especificar propriedades sobre objectos com tempo explícito. A motivação por detrás deste trabalho foi a de desenvolver uma lógica temporal para objectos que fosse decidível e que permitisse raciocinar com o tempo e agregar especificações de objectos.

Para obter decibilidade, foi necessário tomar certas decisões que iriam moldar todo o trabalho: os tipos de dados abstractos foram abandonados recorrendo-se a uma lógica temporal com base proposicional e os operadores temporais X e Y foram consideradas como elementos do alfabeto da lógica.

Para definir assinaturas de objectos sem recorrer a tipos de dados abstractos, optou-se por considerar conjuntos finitos como domínios para os substituir, em vez de considerar acções não parametrizáveis e atributos como expressões booleanas. Os parâmetros das acções e os atributos tomam os valores sobre um conjunto finito, sem comprometer a decibilidade da lógica.

Parte dos operadores temporais utilizados neste trabalho encontram-se definidos em [Koymans 89, Koymans 90]. No entanto, o conceito aí apresentado de tempo relativo, contínuo e pontual foi alterado para permitir ainda a introdução dos operadores temporais X e Y. O tempo foi discretizado, fazendo-se a sua subdivisão em intervalos de comprimento constante. Assim, todos os valores de tempo podem ser utilizados, mas as ocorrências de dois eventos dentro de um mesmo intervalo não se distinguem no que diz respeito ao tempo. O comprimento do intervalo é determinado pelo período do relógio de um objecto. Foi introduzida a definição de máximo divisor comum genaralizado no âmbito da agregação de objectos. Esta aplicação só se encontra definida para números racionais, razão pela qual se tomaram os racionais positivos para domínio dos valores de tempo, sem perda de expressibilidade.

Efectuou-se uma agregação de especificação de objectos respeitando os relógios de cada objecto, com base no co-produto das especificações de objectos. A categoria das assinaturas de objectos e a categoria das especificações de objectos foram ainda apresentadas.

Um dos assuntos em aberto consiste em considerar classes de objectos. Recorrendo a classes de objectos é possível especificar implicitamente vários objectos. A introdução de classes permite considerar heranças de objectos. Diz-se que existe herança de um objecto por parte de um segundo objecto se toda a informação do primeiro se encontra representada no segundo. Com a definição de herança, pode ser introduzida a noção de aspecto. Define-se classe de aspectos e afirma-se que um objecto pode apresentar vários aspectos simultaneamente. Por exemplo, define-se a classe de aspecto `automóvel`. Um objecto da classe seria um automóvel em particular. Com base na classe de aspecto anterior, é possível definir a classe de aspecto `monovolume` que irá herdar toda a informação da classe `automóvel` com mais alguma informação específica. Um objecto da classe `monovolume` terá dois aspectos: o aspecto `automóvel` e o aspecto `monovolume`. Em [Sernadas e Sernadas 94, Sernadas *et al* 95] podemos encontrar uma lógica de especificação de objectos com classes de objectos.

Finalmente seria interessante considerar vários possíveis períodos para o relógio de um objecto. Neste caso, seria necessário considerar uma lógica temporal ramificada, uma vez que existiriam simultaneamente vários futuros possíveis, tantos quanto o número de possíveis períodos.

Outra abordagem interessante e com aplicação imediata consistiria em considerar o tempo como sendo contínuo.

Bibliografia

[Adámek *et al* 90]
J. Adámek, H. Herrlich e G. Strecker. *Abstract and Concrete Categories.* Wiley, 1990.

[Alur e Dill 90]
R. Alur e D. Dill. Automata for modeling real time systems. Em M.S. Paterson (comp), *ICALP 90: Automata, Languages and Programing*, volume 443 de *Lecture Notes in Computer Science*, págs. 322–335. Springer-Verlag, 1990.

[Alur e Henzinge 90]
R. Alur e T. Henzinge. Real-time logics: Complexity and expressiveness. Em IEEE Computer Society Press (comp), *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, págs. 390–401, 1990.

[Alur e Henzinge 91]
R. Alur e T. Henzinge. Time for logic. *SIGACT News*, 22(3):6–12, 1991.

[Barr e Wells 90]
M. Barr e C. Wells. *Category Theory for Computing Science.* Prentice Hall Internacional, 1990.

[Bell e Machover 77]
J.L. Bell e M. Machover. *A Course in Mathematical Logic.* Elsevier Science Publishers B.V., 1977.

[Ben-Hari *et al* 83]
M. Ben-Hari, A. Pnueli e Z. Manna. The temporal logic of branching time. *Acta Informatica*, págs. 207–226, 1983.

[Beth 56]
E. Beth. *Semantic Construction of Instuitionistis Logic.* N. V. North Holland, 1956.

[Böhm 93]
K. Böhm. Real-time object specification logic. Dissertação de Mestrado, University of Darmstadt, Março 1993. Orientada por A. Sernadas.

[Böhm e Sernadas 93]
K. Böhm e A. Sernadas. Real-time object specification logic. Em A. Olivé (comp), *The*

Deductive Approach to Information Systems and Databases, págs. 159–174. Universitat Politécnica de Catalunya, 1993.

[Costa *et al* 89]
J.-F. Costa, A. Sernadas e C. Sernadas. OBLOG *User Manual.* INESC, R. Alves Redol, 9, 1000 Lisboa, 1989.

[Dahl *et al* 67]
O. Dahl, B. Myhrhaug e K. Nygaard. *Simula 67: Common Base Language.* Norwegian Computing Center, 1967.

[Dill 89]
D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. Em J. Sifakis (comp), *CAV 89: Automatic Verification Methods for Finite-state Systems*, volume 407 de *Lecture Notes in Computer Science*, págs. 197–212. Springer-Verlag, 1989.

[Emerson 90]
E. Emerson. Temporal and modal logic. Em J. van Leeuwen (comp), *Handbook of Theoretical Computer Science*, volume B, págs. 995–1072. Elsevier Science Publishers (North-Holland), 1990.

[Felder *et al* 94]
M. Felder, D. Mandrioli e A. Morzenti. Proving proporties of real-time systems through logic specifications and petri net models. *IEEE Transactions on Software Engineering*, 20(2):127–141, Fevereiro 1994.

[Fischer e Ladner 79]
M.J. Fischer e R.E. Ladner. Proposicional dynamic logic os regular programs. *Journal os Computer Systems Science*, 18:194–211, 1979.

[Fitting 83]
M. Fitting. Proof methods for modal and intuitionistic logics. *Synthese Library*, 169, 1983.

[Gallier 87]
J.H. Gallier. *Logic for Computer Science.* Jonh Wiley & Sons, Inc., 1987.

[Goldblatt 87]
R. Goldblatt. *Logics of Time and Computation.* Center for the Study of Language and Information, 1987.

[Gouveia 92]
P. Gouveia. Tableaux para raciocínio local sobre objectos. Dissertação de Mestrado, Universidade Técnica de Lisboa, Outubro 1992. Orientada por C. Sernadas.

[Gouveia *et al* 93]
P. Gouveia, C. Sernadas, J. Gomes e J. Apolinário. Tableaux for reasoning about objects. Em D. Basin, R. Hahnle, B. Fronhofer, J. Possegge e C. Schwind (comps), *Theorem Proving with Analytic Tableaux and Related Methods*, págs. 113–125. Max-Planck-Institut für Informatik, 1993.

[Henzinger 91]
T.A. Henzinger. *The Temporal Specification and Verification of Real-time Systems.* Dissertação de Doutoramento, Stanford University, 1991.

[Hintikka 55]
J. Hintikka. Form and content in quantification theory. *Acta Phil. Fen*, 8:7–55, 1955.

[Jungclaus *et al* 95]
R. Jungclaus, T. Hartmann, G. Saake e C. Sernadas. TROLL: A language for object-oriented specification of information systems. *ACM Transactions on Information Systems*, 13:??–??, 1995.

[Kesten *et al* 93]
Y. Kesten, Z. Manna, H. McGuire e A. Pnueli. A decision algorithm for full propositional temporal logic. Em *Proc. 5th Int. Conf. on Computer-Aided Verification*, volume 697 de *Lecture Notes in Computer Science*, págs. 166–179. Springer-Verlag, 1993.

[Koymans 89]
R. Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic.* Dissertação de Doutoramento, Eindhoven University of Technology, 1989.

[Koymans 90]
R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[Koymans 92]
R. Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic*, volume 651 de *Lecture Notes in Computer Science.* Springer-Verlag, 1992.

[Ladkin 87a]
P. Ladkin. The completeness of a natural system for reasoning with time intervals. Em *Proceedins of International Joint Conference on Artificial Intelligence*, págs. 462–467, Milão, 1987.

[Ladkin 87b]
P. Ladkin. *The Logic of Time Representation.* Dissertação de Doutoramento, University of California, Berkeley, 1987.

[Lee *et al* 94]
I. Lee, P. Brémond-Grégoire e R. Gerber. A process algebraic approach to the specification and analysis of resource-bound real-time systems. *Proceedins of the IEEE*, 82(1):158–185, Janeiro 1994.

[Li e Carmo 93]
R. Li e J. Carmo. On completeness of a positional interval logic with equality, overlap and subinterval relations. *The Bulletin of the Interest Group in Pure Applied Logics*, 1993.

[Li e Sernadas 91]
R. Li e A. Sernadas. Reasoning about objects using tableau method. *Journal of Logic and Computation*, 1(5):575–611, 1991.

[MacLane 71]
S. MacLane. *Categories for the Working Mathematician.* Springer-Verlag, 1971.

[Manna e Pnueli 91]
Z. Manna e A. Pnueli. Completing the temporal picture. *Theoretical Computer Science*, 83:97–130, 1991.

[Manna e Wolper 84]
Z. Manna e P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6:68–93, 1984.

[Ostroff 89]
J. S. Ostroff. *Temporal Logic for Real-time Systems.* Advanced Software Development Series. Research Studies Press Ltd., Inglaterra, 1989.

[Pnueli 77]
A. Pnueli. The temporal logic of programs. Em IEEE Computer Society Press (comp), *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, págs. 46–57, 1977.

[Pnueli e Harel 88]
A. Pnueli e E. Harel. Applications of temporal logics to the specification od real-time systems. Em M. Joseph (comp), *Formal Techiques in real-time and Fault-tolerant Systems*, volume 331 de *Lecture Notes in Computer Science*, págs. 84–98. Springer-Verlag, 1988.

[Pnueli e Sherman 81]
A. Pnueli e R. Sherman. Semantic tableaux for temporal logic. *Technical Report of Computer Science*, 81, 1981.

[Ramos 94]
J. Ramos. Lógica de certificação GNOME. Dissertação de Mestrado, Universidade Técnica de Lisboa, Novembro 1994. Orientada por A. Sernadas.

[Ramos e Sernadas 95a]
J. Ramos e A. Sernadas. A brief introduction to GNOME. Relatório técnico, Secção de Ciência da Computação, Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa, 1995.

[Ramos e Sernadas 95b]
J. Ramos e A. Sernadas. GNOME certification logic. Relatório técnico, Secção de Ciência da Computação, Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa, 1995.

[Resende 91]
P. Resende. Tableaux para lógica dinâmica proposicional. Dissertação de Mestrado, Universidade Técnica de Lisboa, Julho 1991. Orientada por A. Sernadas.

[Sernadas 80]
A. Sernadas. Temporal aspects of logical procedure definition. *Information Systems*, 5:167–187, 1980.

[Sernadas *et al* 89]
A. Sernadas, J. Fiadeiro, C. Sernadas e H.-D. Ehrich. The basic building blocks of information systems. Em E. Falkenberg e P. Lindgreen (comps), *Information Systems Concepts: An In-Depth Analysis*, págs. 225–246. North-Holland, 1989.

[Sernadas *et al* 91]
C. Sernadas, P. Resende, P. Gouveia e A. Sernadas. In-the-large object-oriented design of information systems. Em F. Van Assche, B. Moulin e C. Rolland (comps), *The Object-Oriented Approach in Information Systems*, págs. 209–232. North-Holland, 1991.

[Sernadas *et al* 92a]
C. Sernadas, P. Gouveia, J. Gouveia, A. Sernadas e P. Resende. The reification dimension in object-oriented data base design. Em D. Harper e M. Norrie (comps), *Specification of Database Systems*, págs. 275–299. Springer-Verlag, 1992.

[Sernadas *et al* 92b]
C. Sernadas, P. Gouveia e A. Sernadas. OBLOG: Object-oriented, logic-based conceptual modeling. Relatório técnico, INESC/DMIST, 1992. Submetido para publicação.

[Sernadas *et al* 95]
A. Sernadas, C. Sernadas e J. F. Costa. Object specification logic. *Journal of Logic and Computation*, 5(5):603–630, 1995.

[Sernadas e Ehrich 91]
A. Sernadas e H.-D. Ehrich. What is an object, after all? Em R. Meersman, W. Kent e S. Khosla (comps), *Object Oriented Databases: Analysis, Design and Construction*, págs. 39–69. North-Holland, 1991.

[Sernadas e Ramos 94]
A. Sernadas e J. Ramos. A linguagem GNOME: Sintaxe, semântica e cálculo. Relatório técnico, Secção de Ciência da Computação, Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa, 1994.

[Sernadas e Sernadas 94]
A. Sernadas e C. Sernadas. Object certification. Relatório técnico, Secção de Ciência da Computação, Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa, 1994. Apresentado em DAISD Workshop 94 - Submetido para publicação.

[Sistla e Clarke 85]
A. P. Sistla e E. M. Clarke. The complexity of propositional linear temporal logic. *Journal of the Association for Computing Machinery*, 32(3):733–749, Julho 1985.

[Smullyan 68]
R. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

Apêndice A

Categorias

Neste apêndice introduzimos as principais noções de categorias para esta dissertação. Estas noções encontram-se em [MacLane 71, Adámek *et al* 90, Barr e Wells 90]. Serão também apresentados alguns exemplos de categorias.

A.1 Categorias

Uma categoria consiste num grafo enriquecido com certas propriedades. Um grafo é formado por nós e por setas. As noções de grafos, bem como as noções de categorias são apresentadas em seguida. São também apresentados resultados importantes sobre categorias. No final desta secção, são apresentadas, a título de exemplo, as categorias $\mathcal{S}et$ e $\mathcal{S}et\mathcal{P}nt$.

A.1.1 Conceitos

Definição A.1.1 (Grafo)
Um grafo é um quádruplo $\mathcal{G} = \langle \mathcal{G}_0, \mathcal{G}_1, origem, destino \rangle$ onde:

- $\mathcal{G}_0$ é uma colecção cujos elementos se diz *nós*,
- $\mathcal{G}_1$ é uma colecção cujos elementos se diz *setas*,
- $origem : \mathcal{G}_1 \rightarrow \mathcal{G}_0$ é um mapa que faz corresponder a cada seta um nó, que se designa por *origem* da seta,
- $destino : \mathcal{G}_1 \rightarrow \mathcal{G}_0$ é um mapa que faz corresponder a cada seta um nó, que se designa por *destino* da seta.

$\square$

Definição A.1.2 (Homomorfismo entre Grafos)
Sejam $\mathcal{H}, \mathcal{G}$ dois grafos. Um homomorfismo $\mathcal{F} : \mathcal{G} \to \mathcal{H}$ é um par de aplicações $\mathcal{F}_0 : \mathcal{G}_0 \to \mathcal{H}_0$ e $\mathcal{F}_1 : \mathcal{G}_1 \to \mathcal{H}_1$ tais que:

se $f : m \to n \in \mathcal{G}_1$ então $\mathcal{F}_1(f) : \mathcal{F}_0(m) \to \mathcal{F}_0(n) \in \mathcal{H}_1$.

□

Definição A.1.3 (Categoria)
Uma categoria é um triplo $\mathcal{C} = \langle \mathcal{G}, com, id \rangle$ onde:

- $\mathcal{G} = \langle \mathcal{G}_0, \mathcal{G}_1, origem, destino \rangle$ é um grafo,
- $com : \mathcal{G}_1 \otimes \mathcal{G}_1 \to \mathcal{G}_1$ é um mapa que faz corresponder uma seta à composição de duas setas do grafo,
- $id : \mathcal{G}_0 \to \mathcal{G}_1$ é um mapa que faz corresponder uma seta a cada nó,

que satisfaz as seguintes condições:

- $origem(com(g, f)) = origem(f)$,
- $destino(com(g, f)) = destino(g)$,
- $com(com(h, g), f) = com(h, com(g, f))$,
- $origem(id(A)) = destino(id(A)) = A$,
- se $f : A \to B \in \mathcal{G}_1$ então $com(f, id(A)) = com(id(b), f) = f$.

□

Notação A.1.4 Seja $\mathcal{C} = \langle \mathcal{G}, com, id \rangle$ uma categoria onde $\mathcal{G} = \langle \mathcal{G}_0, \mathcal{G}_2, origem, destino \rangle$ é um grafo. Numa categoria é usual designar os nós por objectos, as setas por morfismos, a origem por domínio e o destino por co-domínio. O conjunto dos objectos da categoria $\mathcal{C}$ é designado por $|\mathcal{C}|$ e o dos morfismos por $Mor(\mathcal{C})$. O mapa com aplicado a f e g é denotado por $f \circ g$ e o mapa id aplicado a A é denotado por id_A.

□

Definição A.1.5 (Objecto Inicial e Terminal)
Um objecto I de uma categoria $\mathcal{C}$ diz-se inicial sse para qualquer objecto A de $\mathcal{C}$ existe um único morfismo f com domínio I e co-domínio A.
Um objecto T diz-se terminal sse para qualquer objecto b de $\mathcal{C}$ existe um único morfismo g com domínio B e co-domínio T.

□

Definição A.1.6 (Produto Binário)
Seja $\mathcal{C}$ uma categoria e A_1, A_2 dois objectos dessa categoria. O produto de A_1 por A_2 em $\mathcal{C}$, representado por $A_1 \otimes A_2$, é um triplo $\langle A, \mu_1, \mu_2 \rangle$ em que:

- A é um objecto de $\mathcal{C}$,
- μ_1 é um morfismo de $\mathcal{C}$ com domínio A e co-domínio A_1,
- μ_2 é um morfismo de $\mathcal{C}$ com domínio A e co-domínio A_2,
- para qualquer objecto B e morfismos $u_1 : B \rightarrow A_1$ e $u_2 : B \rightarrow A_2$ de $\mathcal{C}$, existe um único morfismo $u : B \rightarrow A$ tal que $\mu_i \circ u = u_i$ para $i = 1, 2$.

$$
\begin{tikzcd}
& A \arrow[dl, "\mu_1"'] \arrow[dr, "\mu_2"] & \\
A_1 & & A_2 \\
& B \arrow[ul, "u_1"] \arrow[ur, "u_2"'] \arrow[uu, dashed, "u"'] &
\end{tikzcd}
$$

Figura A.1: Propriedade universal do produto binário.

□

Definição A.1.7 (Co-produto Binário)
Seja $\mathcal{C}$ uma categoria e A_1, A_2 dois objectos dessa categoria. O co-produto de A_1 por A_2 em $\mathcal{C}$, representado por $A_1 \oplus A_2$, é um triplo $\langle A, \imath_1, \imath_2 \rangle$ em que:

- A é um objecto de $\mathcal{C}$,
- $\imath_1$ é um morfismo de $\mathcal{C}$ com domínio A_1 e co-domínio A,
- $\imath_2$ é um morfismo de $\mathcal{C}$ com domínio A_1 e co-domínio A,
- para qualquer objecto B e morfismos $i_1 : A_1 \rightarrow B$, $i_2 : A_2 \rightarrow B$ de $\mathcal{C}$, existe um único morfismo $i : A \rightarrow B$ tal que $i \circ \imath_j = i_j$ para $j = 1, 2$.

□

Definição A.1.8 (Categoria Cartesiana)
Uma categoria $\mathcal{C}$ diz-se cartesiana sse existe o produto de um qualquer número finito dos seus objectos.

□

$$\begin{tikzcd}
 & A \arrow[dd, dashed, "i"] & \\
A_1 \arrow[ur, "i_1"] \arrow[dr, "i1"'] & & A_2 \arrow[ul, "i_2"'] \arrow[dl, "i_2"] \\
 & B &
\end{tikzcd}$$

Figura A.2: Propriedade universal do co-produto binário.

Proposição A.1.9 Uma categoria $\mathcal{C}$ que tenha objecto terminal T e produtos binários é uma categoria cartesiana.

Demonstração:

Por indução no número de objectos no produto,

Base

$i = 0$: O produto zero-ário consiste no objecto terminal T da categoria. As propriedades do objecto terminal são suficientes para verificar as propriedades do produto.

$i = 1$: O produto unário de A é formado pelo par $\langle A, id_A \rangle$. As propriedades de morfismo identidade são suficientes para verificar as propriedades do produto.

$i = 2$: Pelo enunciado, existem sempre produtos binários.

Hipótese Para quaisquer objectos $A_1, \ldots A_i$ com $j \leq n$ existe o produto $\langle A, \mu_1, \ldots, \mu_i \rangle$ em $\mathcal{C}$.

Passo Sejam $A_1, \ldots A_{n+1}$ objectos de $\mathcal{C}$ e $A_1 \times \cdots \times A_n = \langle A', \mu_1, \ldots, \mu_n \rangle$ o produto n-ário de $A_1, \ldots, A_n$. Seja $A' \times A_{n+1} = \langle A, \mu, \mu_{n+1} \rangle$ o produto binário de A' e A_{n+1}. O produto $(n+1)$-ário dos objectos $A_1, \ldots, A_{n+1}$ é definido por $\langle A, \mu_1 \circ \mu, \ldots, \mu_n \circ \mu, \mu_{n+1} \rangle$. Para verificar a propriedade universal do produto, supomos definido um objecto B e morfismos $u_j : B \to A_j$ para $j = 1, \ldots, n+1$. Pela propriedade universal do produto $A_1 \times \cdots \times A_n$, existe um único morfismo $u' : B \to A'$ tal que $\mu_j \circ u' = u_j$ para $j = 1, \ldots, n$. Novamente pela propriedade universal do produto $A' \times A_{n+1}$, existe um único morfismo $u : B \to A$ tal que $\mu \circ u = u'$ e $\mu_{n+1} \circ u = u_{n+1}$. Recorrendo às igualdades anteriores, é possível afirmar que $(\mu_j \circ \mu) \circ u = u_j$ para $j = 1, \ldots, n$ e $\mu_{n+1} \circ u = u_{n+1}$, garantindo a propriedade universal do produto $A_1 \times \cdots \times A_{n+1}$.

$\square$

Definição A.1.10 (Categoria Co-cartesiana)

Uma categoria $\mathcal{C}$ diz-se co-cartesiana sse existe co-produto de um qualquer número finito dos seus objectos.

$\square$

Proposição A.1.11 Uma categoria que tenha objecto inicial e co-produtos binários é uma categoria cartesiana.

Demonstração:

Semelhante à demostração de A.1.9, mas com co-produtos.

$\square$

$$\begin{tikzcd}
 & & A \arrow[dl, "\mu"'] \arrow[ddrr, "\mu_{n+1}"] & & \\
 & A' \arrow[dl, "\mu_1"'] \arrow[dr, "\mu_n"] & & & \\
A_1 & \cdots & A_n & & A_{n+1} \\
 & & B \arrow[ull, "u_1"] \arrow[u, "u_n"'] \arrow[urr, "u^{n+1}"'] \arrow[uul, dashed, "u'"] & &
\end{tikzcd}$$

Figura A.3: Propriedade universal do produto $(n+1)$-ário.

A.1.2 Exemplos

Nesta secção apresentamos as definições de duas categorias: a categoria $\mathcal{S}et$ e a categoria $\mathcal{S}et\mathcal{P}nt$. Será também apresenta a definição de categoria dual.

Prop/Definição A.1.12 (Categoria $\mathcal{S}et$)
A categoria $\mathcal{S}et$ tem como objectos os conjuntos e como morfismos as aplicações entre conjuntos.

$\square$

Na categoria anterior, o objecto inicial é o conjunto vazio e o final é o conjunto singular. O produto de dois conjuntos é formado pelo produto cartesiano dos conjuntos e pelas aplicações de projecção. O co-produto de dois conjuntos é formado pela união disjunta dos conjuntos e pelas aplicações de injecção.

Prop/Definição A.1.13 (Categoria $\mathcal{S}et\mathcal{P}nt$)
A categoria $\mathcal{S}et\mathcal{P}nt$ tem como objectos os pares $\langle C, c\rangle$ onde C é um conjunto e $c \in C$. Os morfismos entre os objectos $\langle C_1, c_1\rangle$ e $\langle C_2, c_2\rangle$ são as aplicações f entre conjuntos C_1 e C_2 tais que $f(c_1) = c_2$.

$\square$

O objecto inicial e final consiste no par $\langle \{c\}, c\rangle$. O produto de $\langle C_1, c_1\rangle$ por $\langle C_2, c_2\rangle$ é formado pelo objecto $\langle C, (c_1, c_2)\rangle$ e pelas projecções π_1, π_2 tais que $\langle C, \pi_1, \pi_2\rangle$ definem o produto em $\mathcal{S}et$ de C_1 por C_2. O co-produto de $\langle C_1, c_1\rangle$ por $\langle C_2, c_2\rangle$ apresenta uma definição mais complicada. Seja $\langle C, \imath_1, \imath_2\rangle$ o co-produto de C_1 e C_2 em $\mathcal{S}et$. Definimos os conjunto C' como sendo

$$C' = C - \{\imath_1(c_1), \imath_2(c_2)\} \cup \{c'\}$$

onde $c' \notin C$. Os morfismos $\imath'_i$ terão a seguinte definição:

$$\imath'_i(c) = \begin{cases} c' & \text{se } c = c_i \\ \imath_i(c) & \text{se } c \neq c_i \end{cases}$$

O co-produto $C_1 \oplus C_2$ é definido pelo triplo $\langle\langle C', c'\rangle, \imath'_1, \imath'_2\rangle$.

As duas categorias anteriores são cartesianas e co-cartesianas.

Definição A.1.14 (Categoria Dual)
Seja $\mathcal{C}$ uma categoria. A categoria dual $\mathcal{C}^{op}$ é definida como se segue:

- $|\mathcal{C}^{op}| = |\mathcal{C}|$,
- se $f : A \to B \in Mor(\mathcal{C})$, então $f : B \to A \in Mor(\mathcal{C}^{op})$,
- se $h = g \circ f \in Mor(\mathcal{C})$, então $h = f \circ g \in Mor(\mathcal{C}^{op})$,
- a identidade de A em $\mathcal{C}^{op}$ é a identidade de A em $\mathcal{C}$.

$\square$

A.2 Functores

Um functor consiste num par de aplicações que traduz objectos e morfismos de uma categoria para os objectos e morfismos de outra categoria, e satisfaz certas propriedades.

Definição A.2.1 (Functor)
Sejam $\mathcal{C}$ e $\mathcal{D}$ duas categorias. Um functor $\mathcal{F} : \mathcal{C} \to \mathcal{D}$ é um par $(\mathcal{F}_0 : \mathcal{C}_0 \to \mathcal{D}_0, \mathcal{F}_1 : \mathcal{C}_1 \to \mathcal{D}_1)$ de mapas tais que:

- se $f : A \to B$ é um morfismo de $\mathcal{C}$, então $\mathcal{F}_1(f) : \mathcal{F}_0(A) \to \mathcal{F}_0(B)$ é um morfismo de $\mathcal{D}$,
- $\mathcal{F}_1(id_A) = id_{\mathcal{F}_0(a)}$ para todo objecto $A \in |\mathcal{C}|$,
- se $f \circ g$ se encontra definido em $\mathcal{C}$, então $\mathcal{F}_1(f) \circ \mathcal{F}_1(g)$ encontra-se definido em $\mathcal{D}$ e $\mathcal{F}_1(f \circ g) = \mathcal{F}_1(f) \circ \mathcal{F}_1(g)$.

$\square$

Prop/Definição A.2.2 (Categoria das Categorias)
A categorias das categorias tem como objectos categorias e como morfismos os functores entre categorias. Esta categoria é designada por $\mathcal{C}at$.

□

Definição A.2.3 (Categoria Cartesiana Livremente Gerada)
Seja $\mathcal{F}cc : \mathcal{P}os \to \mathcal{C}at$ o functor que associa a cada ordem parcial a sua extensão cartesiana canónica. Para a ordem parcial $P = \langle |P|, \leq \rangle$, a categoria $\mathcal{F}cc(p)$, que se denota por $\mathcal{C}_P$, é definida do seguinte modo:

- $|\mathcal{C}_P|$ é tal que $\langle |\mathcal{C}_P|, \otimes, 1 \rangle$ é o monóide livremente gerado por $|P|$,
- $Mor(\mathcal{C}_P)$ é o conjunto quociente induzido no conjunto definido indutivamente como se segue:
  - $(\imath : c' \to c) \in Mor(\mathcal{C}_P)$
    se $c' \leq c$,
  - $(\imath : c \to c) \in Mor(\mathcal{C}_P)$
    se $c \in |\mathcal{C}_P| - |P|$,
  - $(\tau : c \to 1) \in Mor(\mathcal{C}_P)$
    se $c \in |\mathcal{C}_P|$,
  - $(\pi_1 : c \otimes c' \to c) \in Mor(\mathcal{C}_P)$
    se $c, c' \in |\mathcal{C}_P|$,
  - $(\pi_2 : c \otimes c' \to c') \in Mor(\mathcal{C}_P)$
    se $c, c' \in |\mathcal{C}_P|$,
  - $((\sigma' : c' \to c'') \circ (\sigma : c \to c') : c \to c'') \in Mor(\mathcal{C}_P)$
    se $(\sigma' : c' \to c''), (\sigma : c \to c') \in Mor(\mathcal{C}_P)$,
  - $((\sigma' : c' \to c'') \otimes (\sigma'' : c \to c') : c \to c \otimes c'') \in Mor(\mathcal{C}_P)$
    se $(\sigma' : c' \to c''), (\sigma'' : c \to c') \in Mor(\mathcal{C}_P)$,

  pela relação de equivalência induzida pelas equações:

  - $((\sigma : c \to c') \circ (\imath : c \to c) : c \to c') = (\sigma : c \to c')$,
  - $((\imath : c' \to c') \circ (\sigma : c \to c') : c \to c') = (\sigma : c \to c')$,
  - $((\imath : c' \to c'') \circ (\imath : c \to c') : c \to c'') = (\imath : c \to c'')$,
  - $(((\sigma'' : c'' \to c''') \circ (\sigma' : c' \to c'')) \circ (\sigma : c \to c') : c \to c'''') =$
    $((\sigma'' : c'' \to c''') \circ ((\sigma' : c' \to c'') \circ (\sigma : c \to c')) : c \to c'''))$
  - $((\pi_1 : c' \otimes c'' \to c') \circ ((\sigma' : c \to c') \otimes (\sigma'' : c \to c'') : c \to c' \otimes c'')) : c \to c') =$
    $(\sigma' : c \to c')$,

- $((\pi_2 : c' \otimes c'' \to c'') \circ ((\sigma' : c \to c') \otimes (\sigma'' : c \to c'') : c \to c' \otimes c'')) : c \to c'') =$
  $(\sigma'' : c \to c'')$,
- $(((\pi_1 : c' \otimes c'' \to c') \circ (\sigma : c \to c' \otimes c'') : c \to c') \otimes ((\pi_2 : c' \otimes c'' \to c'') \circ$
  $(\sigma : c \to c' \otimes c'') : c \to c'') : c \to c' \otimes c'') = (\sigma : c \to c' \otimes c'')$,
- $(\tau : c \to 1) = (\sigma : c \to 1)$;

- $\imath : c \to c$ é a identidade de c,
- a composição de $\sigma : c \to c'$ com $\sigma' : c' \to c''$ é definida por $((\sigma' : c' \to c'') \circ (\sigma : c \to c') : c \to c'')$.

A $\mathcal{C}_P$ dá-se o nome de categoria cartesiana livremente gerada a partir de P.

□