CÁLCULO DE ACTIVIDADES E TRANSACÇÕES (*)

José Félix Gomes da Costa (**)
Amílcar dos Santos Costa Sernadas (***)

1 — Introdução

Muitos modelos de processos (Brookes *et al,* 1984; Hennessy, 1988; Hoare, 1985; Olderog, 1989), mais ou menos sofisticados, têm vindo a ser propostos salientando as vantagens de cada um dos três domínios semânticos da computação: axiomático, denotacional e operacional. Surgiram, deste modo, contribuições notáveis acerca do não determinismo e de limitações do paradigma de intercalação. Pode mesmo dizer-se que a noção de processo está matematicamente bem fundada. Este estado de coisas é reflectido, por exemplo, no livro de Hennessy (Hennessy, 1988) e na *habilitationsschrift* de Olderog (Olderog, 1989).

Operacionalmente, as equivalências entre processos são definidas em termos da estrutura ramificada dos sistemas de transições associados aos termos de processo sobre uma dada assinatura. As equivalências observam também de maneira diferente o axioma relativo à escolha $e.(p + q) = e.p + e.q$ em virtude da diferença óbvia relativamente à estrutura ramificada. As equivalências distinguem também entre concorrência e não determinismo. Tipicamente estas equivalências observam de maneira diferente o axioma relativo à composição paralela $a.p \parallel b.q = a.(p \parallel b.q) + b.(a.p \parallel q)$ em virtude da diferença óbvia relativamente aos paradigmas de concorrência e *interleaving*. Neste artigo, vamos trabalhar com o mais simples modelo de processos, aceitando os dois últimos axiomas equacionais (este fragmento determinista será designado por modelo de quiescência). O leitor interrogar-se-á: que contribuição pode ainda ser dada à teoria de processos por um modelo tão simples?

Nos conjuntos de traços há informação que pode ser usada para modelar aspectos muito relevantes dos sistemas computacionais. Estes aspectos são conhecidos na área de modelação conceptual orientada por objectos. Nesta disciplina de programação foi proposto que um objecto é um processo munido de atributos dependentes do estado (Ehrich e Sernadas, 1990; Ehrich *et al,* 1990). E também se reconheceu que era necessário demarcar os objectos

---

(*) Prémio Científico IBM, 1992 (menção honrosa).
(**) Faculdade de Ciências da Universidade de Lisboa.
(***) Instituto Superior Técnico.

activos dos objectos passivos (v. Sernadas *et al,* 1990). A maioria dos objectos activos são implementados através de uma linguagem de programação, e suportados pelo gestor de processos do sistema operativo, e muitos dos objectos passivos são implementados através de uma linguagem de esquemas de bases de dados, e suportados pelo gestor de informação de bases de dados. A única diferença intrínseca entre uma componente de arquivo e uma componente de programa está na actividade. O primeiro é passivo e o segundo activo. Quer dizer, o segundo tem requisitos de actividade no sentido em que tem iniciativa para atingir objectivos por si mesmo (por exemplo, terminar a execução de um programa), enquanto o primeiro espera passivamente pelas interacções com os objectos activos vizinhos. Quer dizer, as componentes activas dispõem de tempo de CPU, mas as passivas não.

O modelo de processos activos deterministas obtém-se do modelo tradicional dos traços (modelo PS, de *prefix-closed set*), relaxando o requisito de fecho para prefixos. Consideremos, para aclarar as ideias, uma certa máquina de vendas, MÁQUINA. Trata-se de um processo (cf. Hoare, 1985) denotado em CSP pelo termo:

$$\text{MÁQUINA} = \mu x_{\{\text{moeda, choc}\}}.\ \text{moeda}.\ !\text{choc}.\ x_{\{\text{moeda, choc}\}}$$

A semântica clássica de um tal termo de processo é o par $<E,L>$ com $E = \{\text{moeda, choc}\}$ e $L = (\text{moeda choc})^*(\text{moeda} + \varepsilon)$. Este conjunto de traços não providencia informação acerca da actividade do processo (indicada pela exclamação !) no estado anterior à execução do evento choc, nomeadamente a máquina deverá proceder à entrega de um chocolate. A actividade não é capturada pela semântica clássica de conjuntos fechados para prefixos.

A ideia de quiescência, contribuição de Chandy e Misra (Misra e Chandi, 1981; Misra *et al,* 1982; Misra, 1984) e também desenvolvida por Jonsson (Jonsson, 1985) no contexto de uma lógica para raciocinar acerca de especificações, e por Costa (1989) num contexto algébrico, é uma solução para o problema de representar a actividade dependente do estado de um processo. Dizemos que um processo (ou uma comunidade de processos) está num estado quiescente se não comunica com o seu ambiente a menos que seja este a activar a comunicação. Retomando o exemplo da máquina de vendas, os seus traços quiescentes são, de entre os traços anteriores, aqueles que têm igual número de ocorrências de moeda e de choc, isto é, o conjunto dos traços é agora denotado pela expressão regular (moeda choc)*.

Mas a ideia tem de ser cuidadosamente implantada, se desejamos incluir o tratamento de definições recursivas: os traços infinitos também são quiescentes porque um traço infinito não pode estender-se mais: um RELÓGIO que repete indefinidamente o evento tic:

$$\text{RELÓGIO} = \mu x_{\{\text{tic}\}}.\ !\text{tic}.\ x_{\{\text{tic}\}}$$

não pode encontrar-se num estado quiescente mas pode ser interpretado pela sequência infinita tic^{ω}. Neste artigo vamos considerar primeiro processos que são limite de comportamento finitário. Os requisitos infinitários serão explicados a seguir, no contexto de um modelo semântico mais complexo.

Um processo, ou uma comunidade de processos, é representado pelo conjunto dos seus traços quiescentes. Um traço da comunidade é quiescente se pode ser projectado num traço quiescente de todas as suas componentes, porque a comunidade está num estado quiescente apenas no caso de todas as suas componentes se encontrarem em estados quiescentes. Os requisitos de segurança (que prescrevem, em todo o estado de um dado processo, os eventos que podem ocorrer) podem ser sempre recuperados a partir do fecho para prefixos do conjunto dos traços quiescentes. Os estados activos correspondem aos traços não quiescentes: um traço não quiescente é sempre prefixo de um traço quiescente. O resultado notável que constitui a nossa contribuição é a simplicidade com que é possível descrever a composição paralela na presença de actividade, sendo mesmo possível encontrar um sistema inequacional correcto e adequado para derivar a actividade de uma comunidade, conhecidas as actividades das partes.

De acordo com o modelo que apresentámos intuitivamente, e que será denotado por QS (de *quiescent set*), dado um alfabeto *E*, todo o par $<E,L>$, onde *L* é uma linguagem sobre *E*, é um processo. Por exemplo, o par <{moeda, choc}, (moeda choc)*> é um processo *QS*, mas não é um processo *PS*, uma vez que não é fechado para prefixos. Claramente, todo o processo *PS* é também um processo QS. Tais processos são considerados sem actividade ou passivos, pois encontram-se sempre em estados quiescentes. Os processos *QS* que não são processos *PS* apresentam actividade. O processo apresentado atrás <{moeda, choc}, (moeda choc)*> é activo no sentido em que, por exemplo, após moeda tem iniciativa para realizar choc. Dado um processo QS, $<E,L>$, podemos encontrar o processo passivo correspondente: é $<E, pc(L)>$, onde $pc(L)$ é o fecho para prefixos de *L*.

Dois processos com o mesmo fecho para prefixos são comparáveis relativamente à actividade: quanto menos traços quiescentes, maior é a actividade. Assim, todo o processo tem pelo menos a actividade do seu fecho para prefixos. Esta comparação conduz a uma ordem parcial nos processos.

Para definir o domínio semântico, é necessário enriquecer a linguagem tradicional dos processos deterministas distinguindo duas formas de prefixação: espontânea ou activa (!) e não espontânea ou passiva. Informalmente, se *p* é um termo de processo e *a* é um evento, ambos *a.p* e *!a.p* são termos de processo. Naturalmente, queremos impor que

$$L(a.p) = \{\varepsilon\} \cup \{as\colon s \in L(p)\}$$
$$L(!e.p) = \{as\colon s \in L(p)\}$$

Assim, o processo <{moeda, choc}, (moeda choc)*> atrás referido é denotado, por exemplo, pelo termo de processo:

$$\text{MÁQUINA-ACTIVA} = \mu x_{\{\text{moeda,choc}\}}.\ \text{coin.}\ !\text{choc.}\ x_{\{\text{moeda,choc}\}}$$

e o seu «fecho para prefixos» é denotado por:

$$\text{MÁQUINA-PASSIVA} = \mu x_{\{\text{moeda,choc}\}}.\ \text{moeda.}\ \text{choc.}\ x_{\{\text{moeda,choc}\}}$$

Surpreendentemente, os requisitos de actividade associados ao comportamento transaccional (do tipo «executar até ao fim» podem ser interpretados neste modelo. De facto, podemos representar os requisitos transaccionais não incluindo na linguagem os prefixos de cada transacção. Por exemplo, o processo:

$$\text{MÁQUINA-NOVA} = \mu x.\ \text{moeda.}\ (!\text{café.}\ x + !\text{bolo.}\ x)$$

começa com o requisito transaccional: se ocorrer moeda deverá executar café ou bolo, pois o prefixo moeda de «moeda café» e «moeda bolo» não está no conjunto dos seus traços quiescentes. Isto é, o processo começa com as acções compostas |moeda café| e |moeda bolo|. Sequências finitas de eventos como estas designam-se por transacções.

Um processo $<E,L>$ não tem comportamento transaccional se não tem actividade, isto é, se L é fechado para prefixos. Caso contrário, diz-se que tem requisitos transaccionais. Em qualquer caso, seja $<\text{alf}_T(E),R(L)>$ o resultado de reconverter o processo $<E,L>$, onde $\text{alf}_T(E) = \{|\, e\, |: e \in E^+\}$ é o alfabeto das transacções sobre E e R(L) obtém-se reconvertendo os traços de eventos de E do processo $<E,L>$ em traços de transacções em $\text{alf}_T(E)$. O processo reconvertido envolve uma mudança de granularidade: cada evento pode ser observado como uma acção complexa. Reconhecemos que todo o conjunto L de traços quiescentes sobre E, contendo a sequência vazia ε, pode ser reconvertido pela aplicação de reconversão R: $\mathcal{P}(E^*)\rightarrow\mathcal{P}[\text{alf}_T(E)^*]$, recursivamente definida por:

$$R(L) = \{\varepsilon\} \cup \bigcup_{e \in trM(L)} \{|\, e\, |\, s: s \in R[(r: er \in L)]\},$$

onde $trM(L) = \{e \in L: e \neq \varepsilon \wedge \forall f \in \text{pref}(e)\ (f \in L \Rightarrow (f = \varepsilon \vee f = e)\}$ é o menu transacional de L. O processo que resulta da reconversão de um dado processo exibe a seguinte propriedade: o conjunto dos seus traços está fechado para prefixos, independentemente da actividade do processo original.

Uma transacção (v. Bernstein *et al,* 1987; Cellary *et al,* 1988; Gifford e Donahue, 1985; Gray, 1980) é tratada como a execução de um programa que contém operações sobre uma base de dados com indicadores especiais de

início e fim de transacção. Uma transacção pode terminar normal ou anormalmente. No primeiro caso dizemos que foi bem sucedida ou cometida. No segundo caso dizemos que abortou. Até uma transacção ser cometida pode abortar. Mais: uma transacção tem as seguintes propriedades (v. Cellary *et al,* 1988 para mais detalhes): *a*) Consistência, o que significa que cada transacção conduz um processo de um estado consistente para outro estado também consistente, embora o processo possa não estar em estados consistentes no decurso da execução da transacção; *b*) Atomicidade, o que significa que ou todas ou nenhuma das acções que constituem a transacção devem ser executadas. Por outras palavras, se uma transacção abortar, então têm de ser repostas todas as condições verificadas no início da sua execução, através de algum mecanismo de *roll-back;* c) Durabilidade, o que significa que, quando uma transacção é cometida, as alterações induzidas por ela são irreversíveis mesmo no caso de o sistema ir abaixo. Porém, não obrigamos as transacções a serem independentes umas das outras, isto é, elas podem comunicar. No nosso modelo as transacções podem ser agregadas de modo a sincronizarem em eventos apropriados [uma ideia similar é explorada em Gorrieri e Montanari (1991): «sincronização de transacções é uma transacção»].

No nosso modelo denotacional apenas podemos capturar estados consistentes. Assim, um sistema formal que explique o comportamento transaccional tem de ser suficientemente robusto para reduzir as transacções abortadas em estados inconsistentes ao estado consistente imediatamente anterior: este é o mecanismo de *roll-back.* Para clarificar as ideias, consideremos um freguês da máquina de vendas que, após a inserção da moeda, pretende um café e um bolo:

$$\text{FREGUÊS} = \mu x.\ |\ \text{moeda café bolo}\ |\ .\ x$$

Se realizarmos a composição paralela de MÁQUINA-NOVA e FREGUÊS, como a máquina não pode dispensar ambos *bolo* e *café,* o processo composto alcança um estado inconsistente e tem de retornar ao estado anterior à inserção da moeda (por exemplo, devolvendo a moeda), que é o último estado consistente, isto é:

$$\mu x.\ |\text{moeda café bol}|\ .\ x \parallel \mu x.\ |\text{moeda caf}|\ .\ x + |\text{moeda bol}|\ .\ x = \text{stop}$$

mesmo sabendo que as componentes não atingem um estado de impasse imediatamente após a ocorrência de moeda. É deste modo que modelamos consistência e atomicidade.

A apresentação do modelo — primeiro de processos com actividade e, depois, reconhecendo nestes processos as transacções — segue o esquema supracitado. Na secção 2 introduziremos a semântica denotacional dos processos com requisitos de actividade bem como um sistema formal correcto e

adequado para provar asserções acerca da actividade. Especial relevância será dada à composição paralela, de modo a explicar a actividade das comunidades a partir da actividade dos seus componentes.

Na secção 3 introduziremos as transacções e requisitos transaccionais. Consideraremos apenas transacções lineares que, relativamente ao modelo clássico de transacções (Gray, 1980; Kanellakis, 1980), correspondem a uma relação linear de precedência. Como a actividade pode ser usada para representar transacções, esperamos poder derivar propriedades de processos com transacções observando estes como processos com requisitos de actividade. O sistema formal para raciocinar sobre processos é também correcto e adequado para o cálculo de sistemas comunicantes com transacções. Na secção 4 introduziremos um domínio mais folgado para dar semântica a processos com requisitos de actividade infinitários tal como o relógio atrás introduzido.

2 — Progresso e actividade — modelo QS

Esta secção é dedicada aos processos *QS,* apresentando a linguagem, o domínio denotacional e o sistema de inferência inequacional. Apesar de usarmos técnicas de prova desenvolvidas em Hennessy (1988), adoptamos mais uma vez os conceitos de CSP em Hoare (1985) e Olderog (1991). Claro que o sistema formal e a semântica denotacional constituem contribuição original.

No que se segue E denota um conjunto infinito, fixado à partida.

Definição 2.1 — A *assinatura dos processos deterministas com actividade* é $\Sigma_{QS} = \Sigma_{QS}^0 \cup \Sigma_{QS}^1 \cup \Sigma_{QS}^2$ com $\Sigma_{QS}^0 = (\mathrm{stop}_U, \mathrm{abort}_U : U \in \mathcal{P}_f(E))$, $\Sigma_{QS}^1 = \{e. : e \in E\} \cup \{!e. : e \in E\}$ e $\Sigma_{QS}^2 = \{+, \|\}$.

Cada Σ_{QS}^j com $j=0..2$, é o conjunto dos símbolos de operação de aridade j. Como é usual, escrevemos $e.p$ e $!e.p$. ao invés de $e.(p)$ e $!e.(p)$, respectivamente. Além do mais, assumimos que a prefixação tem prioridade sobre a escolha. Seja $\Sigma_{QS}\dagger = \Sigma_{QS} \setminus \{\|\}$.

Fixa-se à partida uma família de conjuntos infinitos totalmente ordenados contáveis: $(X_U)_{U \in \mathcal{P}_f(E)}$ (*). Os elementos de cada X_U dizem-se *variáveis* (de processo) de alfabeto U. Assumimos que estes conjuntos são disjuntos dois a dois e disjuntos de E. x_U, y_U e z_U designam variáveis genéricas em X_U. x, y e z designam variáveis genéricas em $X = \cup_{U \in \mathcal{P}_f(E)} X_U$.

Definição 2.2. — O conjunto dos *termos recursivos* sobre X relativos à assinatura Σ_{QS}, que se denota por $\mathrm{Rec}_{QS}(X)$, é o menor conjunto que satisfaz *a*) se $x \in X$ então $x \in \mathrm{Rec}_{QS}(X)$; *b*) se $f \in \Sigma_{QS}^k$ e $t_1, \ldots, t_k$ são termos de $\mathrm{Rec}_{QS}(X)$, então $f(t_1,\ldots,t_k) \in \mathrm{Rec}_{QS}(X)$, e *c*) se $x \in X$ e $t \in \mathrm{Rec}_{QS}(X)$ *então* $\mu x.t \in \mathrm{Rec}_{QS}(X)$.

(*) $P(X)$ denota o conjunto das partes de X e $P_f(X)$ denota o conjunto das partes finitas de X.

Seja $\mathrm{Rec}_{QS}\dagger(X)$ o conjunto dos termos recursivos sobre $\Sigma_{QS}\dagger$. O conjunto dos *termos finitos* sobre X relativos à assinatura Σ_{QS} é o menor conjunto que satisfaz as alíneas *a*) e *b*) da definição precedente.

A cada termo p em $\mathrm{Rec}_{QS}(X)$ associamos o alfabeto $\mathrm{alf}(p)$ definido por indução: *a*) $\mathrm{alf}(\mathrm{stop}_U) = \mathrm{alf}(\mathrm{abort}_U) = \mathrm{alf}(x_U) = U$; *b*) $\mathrm{alf}(e.p) = \mathrm{alf}(!e.p.) = \{e\} \cup \mathrm{alf}(p)$; *c*) $\mathrm{alf}(p + q) = \mathrm{alf}(p \,||\, q) = (\mathrm{alf}(p) \cup \mathrm{alf}(q)$, e *d*) $\mathrm{alf}(\mu x.p) = \mathrm{alf}(x) \cup \mathrm{alf}(p)$.

Definição 2.3 — A cada termo p em $\mathrm{Rec}_{QS}(X)$ associamos o conjunto dos seus *subtermos* definido indutivamente do modo seguinte: *a*) p é um subtermo de p; *b*) se p é $e.q$ ou $!e.q$, então todo o subtermo de q é um subtermo de p; *c*) se p é $p_1 + p_2$ ou $p_1 || p_2$, então todo o subtermo de p_1 e todo o subtermo de p_2 é subtermo de p, e *c*) se p é $\mu x.p$, então todo o subtermo de q é um subtermo de p. O conjunto dos subtermos de p denota-se por $\mathrm{sub}(p)$.

Definição 2.4 — Seja p em $\mathrm{Rec}_{QS}(X)$ um termo recursivo. O conjunto das *variáveis livres* de p, que se denota por $\mathrm{fi}(p)$ é definido como segue: *a*) $\mathrm{fi}(x) = \{x\}$; *b*) $\mathrm{fi}(\mathrm{stop}_U) = \mathrm{fi}(\mathrm{abort}_U) = \emptyset$; *c*) $\mathrm{fi}(e.p) = \mathrm{fi}(!e.p) = \mathrm{fi}(p)$; *d*) $\mathrm{fi}(p+q) = \mathrm{fi}(p||q) = \mathrm{fi}(p) \cup \mathrm{fi}(q)$, e *e*) $\mathrm{fi}(\mu x.p) = \mathrm{fi}(p) \setminus \{x\}$.

Dizemos ainda que uma variável x *ocorre livre* em $p \in \mathrm{Rec}_{QS}(X)$ se $x \in \mathrm{fi}(p)$; caso contrário, diz-se que x é uma *variável muda.* Um termo p em $\mathrm{Rec}_{QS}(X)$ diz-se fechado se $\mathrm{fi}(p) = \emptyset$.

No seguimento, por $c\mathrm{Rec}_{QS}(X)$ denotamos o subconjunto dos *termos fechados* de $\mathrm{Rec}_{QS}(X)$, por $f\mathrm{Rec}_{QS}(X)$ denotamos o subconjunto dos *termos finitos* de $\mathrm{Rec}_{QS}(X)$ e por $cf\mathrm{Rec}_{QS}(X)$ denotamos o subconjunto dos *termos finitos fechados* de $\mathrm{Rec}_{QS}(X)$. Para denotar termos em $\mathrm{Rec}_{QS}(X)$ ou $c\mathrm{Rec}_{QS}(X)$ usamos os identificadores $p, q, r, \ldots$ e para denotar termos em $f\mathrm{Rec}_{QS}(X)$ ou $cf\mathrm{Rec}_{QS}(X)$ usamos os identificadores $\pi, \chi, \theta, \ldots$

De entre os termos de $\mathrm{Rec}_{QS}(X)$ há termos notáveis para a teoria que vamos desenvolver: são os *termos guardados,* definidos de acordo com a:

Definição 2.5 — Um termo p de $\mathrm{Rec}_{QS}(X)$ diz-se *guardado* se em todo o subtermo recursivo $\mu x.q$ de p toda a ocorrência livre de x em q, ocorre num subtermo da forma $e.r$ ou $!e.r$ de q. Um termo p de $\mathrm{Rec}_{QS}(X)$ diz-se *passivamente guardado* se em todo o subtermo recursivo $\mu x.q$ de p toda a ocorrência livre de x em q, ocorre um subtermo da forma $a.r$ de q.

Definição 2.6 — Um termo de processo é um termo p em $\mathrm{Rec}_{QS}(X)$ que satisfaz as seguintes dependências de contexto: *a*) p é passivamente guardado; *b*) todo o subtermo $e.q$ ou $!e.q$ de p satisfaz $e \in \mathrm{alf}(q)$; *c*) todo o subtermo $q + r$ de p satisfaz $\mathrm{alf}(q) = \mathrm{alf}(r)$, e *d*) todo o subtermo $\mu x.q$ de p satisfaz $\mathrm{alf}(x) = \mathrm{alf}(q)$.

O conjunto dos termos de processo sobre X denota-se por $\mathrm{Proc}(X)$. Por $c\mathrm{Proc}(X)$ denota-se o subconjunto dos termos fechados de $\mathrm{Proc}(X)$, por $f\mathrm{Proc}(X)$ denotamos o subconjunto dos termos finitos de $\mathrm{Proc}(X)$ e por $cf\mathrm{Proc}(X)$ denotamos o subconjunto dos termos finitos fechados de $\mathrm{Proc}(X)$. Seja $\mathrm{Proc}\dagger(X)$ o subconjunto de $\mathrm{Proc}(X)$ de termos de processo também em $\mathrm{Rec}_{QS}\dagger(X)$ (sempre que necessário usaremos também a notação prefixa c e f).

Note-se, relativamente à definição precedente, que o termo $\mu x.x$ não é guardado e, consequentemente, não é um termo de processo. O termo $\mu x.$!tic. x não é passivamente guardado e também não é, por isso, um termo de processo. Mas o termo $\mu x.$ moeda. !choc. x é passivamente guardado porque a variável x ocorre no subtermo moeda. (!choc. x) do termo recursivo $\mu x.$ moeda. !choc. x.

Definição 2.7 — Uma semântica denotacional para termos sobre uma assinatura Σ é uma aplicação $D[\text{-}]$; $\text{Rec}_\Sigma(X) \rightarrow (\text{ENV}_D \rightarrow |D|)$ tal que: 1) o domínio semântico D encontra-se munido de uma ordem parcial $\leq$ e, para cada $A \subset E$, existe um subdomínio $D_A \subset D$ que é uma ordem parcial completa relativamente à restrição de $\leq$ *a* esse subdomínio; 2) o *conjunto de ambientes* ENV_D consiste num conjunto de atribuições $\rho: X \rightarrow |D|$ que respeita alfabetos, isto é, se $\text{alf}(x) = A$, então $\rho(x) \in D_A$; 3) para todo o símbolo $f \in |\Sigma|$ de aridade n, existe uma operação (função) semântica correspondente $f_D: |D|^n \rightarrow |D|$ que é contínua relativamente a $\leq$; 4) a definição de $D[\text{-}]$ procede agora indutivamente: *a*) $D[x]\rho = \rho(x)$; *b*) $D[f(t)]\ \rho = f_D(D[t]\rho)$, e *c*) $D[\mu x.p]\rho = \gamma\Phi_{p,\rho}$ onde $\gamma\Phi_{p,\rho}$ denota o mais pequeno ponto fixo da aplicação $\Phi_{p,\rho}$: $D_{\text{alf}}(x) \rightarrow D_{\text{alf}}(x)$ definida por $\lambda\xi.D[p]\rho[\xi/x])$, onde $\rho[\xi/x])$ denota um ambiente ρ modificado em x, variável à qual atribui o valor de ξ.

Relativamente a um modelo D, dois termos de processo p e q dizem-se iguais se denotam o mesmo processo: $D[p]\rho = {}_A D[q]\rho$, para todo o ambiente ρ. Mais, p diz-se mais espontâneo que q, relativamente a D, se $D[p]\rho \leq {}_D D[q]\rho$, para todo o ambiente ρ. Isto é, $\leq_D$ ordena parcialmente os processos D de acordo com a sua actividade. Se, para todo o ambiente ρ se tem $D[p]\rho \leq_D D[q]\rho$, então escrevemos $\models_D p \leq q$.

Definição/Proposição 2.8 — A álgebra $\text{-}\Sigma_{QS}$, $QS = \langle |QS|, \leq_{QS}, \Sigma_{QS} \rangle$ tem os seus elementos definidos como segue:

$$|QS| = \{\langle E,L\rangle : E \in \mathcal{P}_f(E) \text{ e } L \in \mathcal{P}(E^*)\}$$

$$\langle E,L\rangle \leq_{QS} \langle E'L'\rangle \text{ se } E = E' \text{ e } L' \subseteq L'$$

$$\Sigma_{QS} = \{\text{stop}^{U}_{QS}, \text{abort}^{U}_{QS}: U \in \mathcal{P}_f(E)\} \cup \{e._{QS},\ !e.QS: e \in E\} \cup \{+_{QS}, ||_{QS}\}$$

abort^{U}_{QS}	$: \rightarrow \lvert QS\rvert$	abort^{U}_{QS}	$= \langle U,\varnothing\rangle$
stop^{U}_{QS}	$: \rightarrow \lvert QS\rvert$	stop^{U}_{QS}	$= \langle U,\{\varepsilon\}\rangle$
$e._{QS}$	$: \lvert QS\rvert \rightarrow \lvert QS\rvert$	$e._{QS}(\langle U,L\rangle)$	$= \langle \{e\}\cup U, \{\varepsilon\}\cup\{es\}: s \in L\rangle$
$!e._{QS}$	$: \lvert QS\rvert \rightarrow \lvert QS\rvert$	$!e._{QS}(\langle U,L\rangle)$	$= \langle \{e\}\cup U, \{es: s \in L\}\rangle$
$+_{QS}$	$: \lvert QS\rvert\ \lvert QS\rvert \rightarrow \lvert QS$	$+_{QS}(\langle U,L\rangle, \langle U'L'\rangle)$	$= \langle U\cup U', L\cup L'\rangle$
$\lvert\rvert_{QS}$	$: \lvert QS\rvert\ \lvert QS\rvert \rightarrow \lvert QS$	$\lvert\rvert_{QS}(\langle U,L\rangle, \langle U'L'\rangle)$	$= \langle U\cup U', L \lvert\rvert L'\rangle$

Por $L \,||\, L'$ denotamos, é claro, a linguagem $\{s \in (U\cup U')^*: s{\downarrow}U \in L \text{ e } s{\downarrow}U' \in L'\}$. Dado um processo $P = \langle E,L\rangle$, referimo-nos a E por $\text{alf}(P)$ — o alfabeto de P — e L por $\tau(P)$ — os traços quiescentes de P.

A ordem parcial de actividade estrita introduz-se do seguinte modo:

$$<E,L> \leq_! <E',L'> \text{ se } E = E',\ pc(L) = pc(L') \text{ e } L \subseteq L'$$

isto é, $<E,L> \leq_! <E',L'>$ se $<E,L> \leq_{QS} <E',L'>$, e $pc(L) = pc(L')$. Adiante mostraremos como se pode raciocinar em termos de $\leq_!$ embebida em $\leq_{QS}$. Mas adiantemos já que a ordem parcial $\leq_{QS}$ é já um resultado de sobrepor a ordem parcial indutora do reticulado à ordem parcial de actividade estrita.

A prova de que o domínio semântico é um domínio de Scott não oferece novidade: $<|QS|,\leq_{QS},\Sigma_{QS}>$ é um domínio-Σ_{QS}, sendo $<|QS|_U,\leq_{QS}>$ uma ordem parcial algébrica para cada $U \subseteq E$ não vazio. Também não é difícil a prova da seguinte proposição:

Proposição 2. — O domínio *QS* é finitário, isto é, todo o termo (sintacticamente) finito é interpretado por um elemento finito de *QS* e todo o elemento finito de *QS* é denotável por um termo (sintacticamente) finito.

Prova: É imediato reconhecer que todo o termo (sintacticamente) finito em $Rec_{QS}(X)$ denota um elemento finito de *QS*. Reciprocamente, tomemos um elemento finito arbitrário de *QS* e encontremos um termo (sintacticamente) finito que o denote. Seja $<E,L> \in |QS|$ com *L* finito. Isto é $L = \{s_1\} \cup \ldots \cup \{s_k\}$ com $k \geq 0$. Se $k = 0$ podemos tomar o termo finito de processo $abort_U$. Caso contrário, assumindo que para cada s_i temos um termo de processo p_i de alfabeto *E* tal que $QS[p_i] = <E,\{s_i\}>$, podemos concluir a prova exibindo o termo de processo $p_1 + \ldots + p_k$ que denota $<E,L>$. Assim, só temos de mostrar que, para todo o $s \in E^*$, existe um termo de processo *p* de alfabeto *E* tal que $QS[p] = <E,\{s\}>$. Seja v: $E^* \rightarrow Rec_{QS}(X)$ uma aplicação indutivamente definida como segue: *a*) $v(\varepsilon) = stop U$, e *b*) $v(as) = !a.v(s)$. É trivial reconhecer por indução no comprimento de *s* que $QS\ [v(s)] = <E,\{s\}>$.

Passamos agora a examinar a caracterização sintáctica de *QS* através de um sistema formal correcto e adequado. A ideia consiste em identificar um conjunto *QS* de axiomas próprios tal que $\vdash_{QS} p \leq q$ se $\models_{QS} p \leq q$, quaisquer que sejam os termos de processo fechados *p* e *q*. O resultado das nossas investigações encontra-se condensado no quadro I, onde assumimos que $a \neq b$ sempre que num axioma-esquema surgem *a* e *b*.

Seja $QS^\dagger$ o subsistema de *QS* que não contém as equações relativas à composição paralela. Os sistemas *QS* e *PS* distinguem-se essencialmente em virtude de o axioma $a.\ x_U + stop_{U \cup \{a\}} = a.\ x_U$ ter sido substituído pelo axioma relativo à actividade.

É fácil mas moroso verificar que o sistema QS é redutivo em relação a $QS^\dagger$, isto é, para todo o termo de processo $p \in cfProc(X)$ existe um termo de processo $\chi \in cfProc^\dagger(X)$ tal que $\vdash_{QS} p = \chi$. Para todo o termo $p \in cProc(X)$, podemos encontrar uma forma pseudonormal $\psi(p)$ tal que $\vdash QS\ p = \psi(p)$: este resultado segue do axioma de recursividade e da garantia de que os termos

de processo são guardados. Assim, as equações relativas à composição paralela podem ser manipuladas de modo a transformar todo o termo em cProc(X) numa forma pseudonormal: um termo de processo que tem um símbolo de prefixação ou a escolha como operação mais externa na árvore sintáctica abstracta.

A distributividade da prefixação activa em relação à escolha pode ser derivada dos axiomas, o mesmo se passando com o *teorema da terminação de roll-back:*

Proposição 2.10 — $\vdash_{QS} a.\ \mathrm{abort}_U = \mathrm{stop}_{U\cup\{a\}}$.

QUADRO I

Axiomatização QS

Estrutura

$x_U + x_U$	=	x_U	QS_1
$x_U + y_U$	=	$y_U + x_U$	QS_2
$x_U + (y_U + z_U)$	=	$(x_U + y_U) + z_U$	QS_3
$x_U + \mathrm{abort}_U$	=	x_U	QS_4
$!a.(x_U + y_U)$	=	$!a.x_U + !a.y_U$	QS_5

Reticulado

abort_U	$\leq$	x_U	QS_6

Roll-back

$!a.\mathrm{abort}_U$	=	$\mathrm{abort}_{U\cup\{a\}}$	QS_7

Actividade

$!a.x_U + \mathrm{stop}_{U\cup\{a\}}$	=	$a.x_U$	QS_8

Paralelismo

$(y_V + z_V) \parallel x_U$	=	$y_V \parallel x_U + z_V \parallel x_U$	QS_9
$x_U \parallel y_U$	=	$y_U \parallel x_V$	QS_{10}
$x_U \parallel \mathrm{abort}_V$	=	$\mathrm{abort}_{U\cup V}$	QS_{11}
$!a.x_U \parallel \mathrm{stop}_{V\setminus\{a\}}$	=	$!a.(x_U \parallel \mathrm{stop}_{V\setminus\{a\}})$	QS_{12}
$!a.x_U \parallel \mathrm{stop}_{V\cup\{a\}}$	=	$\mathrm{abort}_{U\cup V\cup\{a\}}$	QS_{13}
$!a.x_{U\setminus\{a\}} \parallel !b.\ y_{V\setminus\{a\}}$	=	$!a.(x_{U\setminus\{b\}} \parallel !b.y_{V\setminus\{a\}})$	
	+	$!b.\ (!a.x_{U\setminus\{b\}} \parallel y_{V\setminus\{a\}})$	QS_{14}
$!a.x_{U\cup\{b\}} \parallel !b.y_{V\cup\{a\}}$	=	$\mathrm{abort}_{U\cup V\cup\{a,b\}}$	QS_{15}
$!a.x_{U\cup\{b\}} \parallel !b.y_{V\setminus\{a\}}$	=	$!a.(x_{U\cup\{b\}} \parallel !b.y_{V\setminus\{a\}})$	QS_{16}
$!a.x_U \parallel !a.y_V$	=	$!a.(x_U \parallel y_V)$	QS_{17}

Prova: Empregando o axioma de actividade, podemos reescrever o termo de processo $a.\ \mathrm{abort}_U$ na forma $!a.\ \mathrm{abort}_U + \mathrm{stop}_{U\cup\{a\}}$. Por *roll-back* origina o termo $\mathrm{abort}_{U\cup\{a\}} + \mathrm{stop}_{U\cup\{a\}}$. Por QS_4, vem que $\vdash_{QS} a.\ \mathrm{abort}_U = \mathrm{stop}_{U\cup\{a\}}$.

E também o *teorema de actividade:*

Proposição 2.11 — $\vdash_{QS} !a.\ x_U \leq a.\ x_U$. QS_{18}

Prova: Empregando axiomas estruturais, obtemos $\vdash_{QS} \mathrm{abort}_{U\cup\{a\}} \leq a.\ x_U$. É então fácil de estabelecer que $\vdash_{QS} \mathrm{abort}_{U\cup\{a\}} + !a.x_U \leq a.\ x_U + !a.\ x_U$. Deste teorema segue-se o teorema enunciado da proposição, empregando o axioma de actividade e axiomas estruturais.

Como exemplo apresentamos a expansão completa do termo de processo $a.\ !b.\ !c.\ \mathrm{stop}_E \parallel a.\ e.\ f.\ \mathrm{stop}_F$ com $E = \{a, b, c\}$ e $F = \{a, e, f\}$. Temos que o resultado da expansão é:

$a.\ !b.\ !c.\ e.\ f.\ \mathrm{stop}_{E\cup F} +$	e, f não activados
$a.\ !b.\ !e.\ !c.\ f.\ \mathrm{stop}_{E\cup F} +$	activação de e; f não activado
$a.\ !b.\ !e.\ !f.\ !c.\ \mathrm{stop}_{E\cup F} +$	activação de e, f
$a.\ !e.\ !f.\ !b.\ !c.\ \mathrm{stop}_{E\cup F} +$	activação de e, f
$a.\ !e.\ !b.\ !c.\ f.\ \mathrm{stop}_{E\cup F} +$	activação de e; f não activado
$a.\ !e.\ !b.\ !f.\ !c.\ \mathrm{stop}_{E\cup F}$	activação de e, f

Note-se que alguns eventos da segunda componente $a.\ e.\ f.\ \mathrm{stop}_F$ têm de ser activados em determinados subtermos da expansão, como acontece com e ou f em $a.\ !e.\ !f.\ !b.\ !c.\ \mathrm{stop}_{E\cup F}$. Isto significa que *na composição paralela não podemos intercalar um evento não espontâneo de uma componente entre dois eventos espontâneos de outra componente sem primeiro o activar.*

A correcção de *QS* verifica-se sem muito trabalho:

Proposição 2.12 — Para quaisquer termos de processo $p, q \in \mathrm{Proc}(X)$, se $\vdash_{QS} p \leq q$, então $\models_{QS} p \leq q$.

Prova: A prova decorre em três etapas: *a*) os axiomas específicos de *QS* são correctos; *b*) o sistema de inferência de *QS* é correcto, e *c*) toda a derivação inequacional de *QS*, empregando as regras do sistema formal, é correcta. As provas de *a*) e de *c*) podem adaptar-se (Hennessy, 1988) e a correcção dos axiomas específicos é rotineira. Comentamos apenas a distributividade da composição paralela em relação à escolha. Este axioma traduz-se na distributividade da intercalação de linguagens relativamente à união. Porém, este facto é verdadeiro apenas no caso de se tratar de união de linguagens relativas ao mesmo alfabeto, tal como acontece no caso corrente (v. Van Snepscheut 1985).

O sistema formal é também adequado relativamente a *QS*, para termos de processo em $\mathrm{cProc}(X)$. Usamos de novo formas normais de modo a provar: a) todo o termo de processo em $\mathrm{cfProc}^{\dagger}(X)$ admite uma forma normal

equivalente, isto é, para todo o termo de processo $p \in \mathrm{cfProc}^{\dagger}(X)$ existe uma forma normal $n(p)$ tal que $\vdash_{QS^{\dagger}} p = n(p)$, e *b*) quaisquer que sejam as formas normais n e m, se $\models_{QS} n \leq m$ então $\vdash_{QS^{\dagger}} n \leq m$. Usando estes resultados segue-se que, quaisquer que sejam os termos de processo p, $q \in \mathrm{cfProc}(X)$, se $\models_{QS} p \leq q$ então $\vdash_{QS} p \leq q$.

A associatividade da escolha e da composição paralela permite omitir parênteses nos termos de processo sem introduzir ambiguidade, tal como temos feito até aqui. A comutatividade e a idempotência permitem também introduzir metatermos de grande conveniência notacional. Se $\Xi = \{p_1, \ldots, p_k\}$ é um conjunto finito de termos de alfabeto U, então denotamos o termo $p_1 + \ldots + p_k$ pelo metatermo $+^{U}_{p \in \Xi}\ p$. Se Ξ é vazio $+^{U}_{p \in \Xi}\ p$ denota simplesmente o termo abort_U.

Definição 2.13 — O conjunto dos termos normais de alfabeto U define-se indutivamente como segue: *a*) stop_U é um termo normal de alfabeto U, e *b*) se E é um subconjunto de U e $n_U(e)$ é um termo normal de alfabeto U, para todo o $e \in E$, então $+^{U}_{e \in E}\ e.n_U(e)$ e $+^{U}_{e \in E}\ !e.n_U(e)$ são termos normais de alfabeto U.

Seja $\mathrm{NProc}_U(X)$ o conjunto dos termos normais de alfabeto U e $\mathrm{NProc}(X) = \cup_U \mathrm{NProc}_U(X)$. O teorema dos termos normais seguinte prova-se por indução na estrutura dos termos:

Proposição 2.14 — Para todo o termo de processo p em $\mathrm{cfProc}^{\dagger}(X)$ tem-se $\vdash_{QS} p = \mathrm{abort}_{\mathrm{alf}(p)}$ ou existe um termo normal n(p) tal que $\vdash_{QS^{\dagger}} p = n(p)$.

O principal resultado relativo à adequação segue:

Proposição 2.15 — Quaisquer que sejam os termos de processo em $\mathrm{cProc}(X)$, se $\models_{QS} p \leq q$ então $\vdash_{QS} p \leq q$.

Prova: Como QS é redutivo em relação a $QS^{\dagger}$, resulta que temos apenas que provar o teorema para termos de processo finitos, isto é, quaisquer que sejam os termos de processo p, q em $\mathrm{cfProc}^{\dagger}(X)$, se $\models_{QS} p \leq q$ então $\vdash_{QS^{\dagger}} p \leq q$. Suponhamos pois que $\models_{QS} p \leq q$. Se $\vdash_{QS} p = \mathrm{abort}_{\mathrm{alf}(p)}$ ou $\vdash_{QS} q = \mathrm{abort}_{\mathrm{alf}(q)}$ o teorema é imediato a partir do axioma de reticulado, ou por reflexividade. Caso contrário, nem p nem q são redutíveis a abort, e $\vdash_{QS^{\dagger}} p = n(p)$ e $\vdash_{QS^{\dagger}} q = n(q)$. Como QS satisfaz todos os axiomas de $QS^{\dagger}$, podemos escrever $\models_{QS} p = n(p)$ e $\models_{QS} q = n(q)$ e, portanto, $\models_{QS} n(p) \leq n(q)$. Provamos agora que $\models_{QS} n(p) \leq n(q)$ implica $\vdash_{QS^{\dagger}} n(p) \leq n(q)$ donde segue imediatamente que $\vdash_{QS^{\dagger}} p \leq q$.

Se $n(p)$ ou $n(q)$ é stop_U, com $U = \mathrm{alf}(p) = \mathrm{alf}(q)$, o teorema segue por reflexividade ou por indução na estrutura de um termo normal. Podemos assumir que $n(p)$ e $n(q)$ são, respectivamente, 1) $+^{U}_{e \in A}\ e.k_U(e)$ e $+^{U}_{e \in B}\ e.n_U(e)$; 2) $+^{U}_{e \in A}\ !e.k_U(e)$ e $+^{U}_{e \in B}\ !e.n_U(e)$; ou 3) $+^{U}_{e \in A}\ !e.k_U(e)$ e $+^{U}_{e \in B}\ e.n_U(e)$. Em todos estes casos temos que $A \subseteq B$ e $\models_{QS} k_U(e) \leq n_U(e)$ para todo o $e \in A$. Aplicando a hipótese de indução, obtemos $\vdash_{QS^{\dagger}} k_U(e) \leq n_U(e)$. Tomando em consideração apenas o caso 2) — nos outros casos a prova é similiar —, obte-

mos $\vdash_{QS}^{\dagger} + \overset{U}{_{e\in A}}\ !e.k_U(e) \leq + \overset{U}{_{e\in A}}\ !e.n_U(e)$. Adicionando abort_U a ambos os termos e aplicando a instância $\vdash_{QS}^{\dagger} \text{abort}_U \leq + \overset{U}{_{e\in B\setminus A}}\ !e.n_U(e)$ do axioma de reticulado, derivamos o resultado pretendido $\vdash_{QS}^{\dagger} n(p) \leq n(q)$.

A adequação relativamente à ordem parcial estrita de actividade é uma propriedade mais interessante da lógica. Seja *rQS* o sistema que se obtém de *QS* retirando o axioma de reticulado e incluindo o teorema QS_{18}. Enunciamos sem demonstrar.

Proposição 2.16 — Quaisquer que sejam os termos de processo p, $q\in \text{cProc}(X)$, se $\models_{QS} p \leq !q$ então $\vdash_{rQS} p \leq q$.

Podemos acelerar a prova de muitos teoremas juntando aos axiomas alguns teoremas triviais mas de grande aplicabilidade na expansão do combinador ||. Exemplos de axiomas-esquema interessantes são:

Proposição 2.17 — $\vdash_{QS} !a.\ x_U \parallel b.\ y_V = !a.\ (x_U \parallel b.\ y_V) + !b.\ (!a.\ x_U \parallel y_V)$, se $a\notin V$ e $b\notin U$.

Prova: Trivial. V. prova da proposição seguinte.

Proposição 2.18 — Se $a\in V$ e $b\in U$ então $\vdash_{QS} !a.\ x_U \parallel b.\ y_V = \text{abort}_{U\cup V}$.

Prova: Empregando o axioma de actividade, podemos reescrever o termo de processo $!a.\ x_U \parallel b.\ y_V$ na forma $!a.\ x_U \parallel (!b.\ y_V + \text{stop}_{V\cup\{b\}})$. Aplicando distributividade, obtemos o termo $!a.\ x_U \parallel !b.\ y_V + !a.\ x_U \parallel \text{stop}_{V\cup\{b\}}$; os axiomas de paralelismo podem agora ser usados de modo a se obter o teorema $\vdash_{QS} !a.\ x_U \parallel b.\ y_V = \text{abort}_{U\cup V} + \text{abort}_{U\cup V} = \text{abort}_{U\cup V}$.

A equação da última proposição mostra que, se uma componente de uma comunidade atinge um estado final inconsistente, então o processo global aborta e retorna ao último estado consistente global. Na secção em que vamos agora entrar veremos o papel deste mecanismo de *roll-back* no cálculo.

3.3 — Transacções e requisitos transaccionais — modelo TP

Debrucemo-nos agora sobre os processos com transacções (sequências de eventos cuja execução uma vez iniciada deve ser conduzida até ao fim) como aplicação da teoria prévia. Como na secção anterior, E denota um alfabeto fixo.

Definição 3.1 — A *assinatura dos processos deterministas com requisitos transaccionais* é $\Sigma_{TP} = \Sigma_{TP}^0 \cup \Sigma_{TP}^1 \cup \Sigma_{TP}^2$ com $\Sigma_{TP}^0 = \{\text{stop}_U\colon U\in \mathcal{P}_f(E)\}$, $\Sigma_{TP}^1 = \{!el.\colon e\in E^+\}$ e $\Sigma_{TP}^2 = \{+, \parallel\}$.

Observe-se que estamos a usar identificadores do alfabeto latino, para denotar sequências finitas não vazias de elementos de E. O conjunto dos termos recursivos sobre Σ_{TP} com variáveis em X denota-se por $\text{Rec}_{TP}(X)$.

O alfabeto de um termo $p\in \text{Rec}_{TP}(X)$ define-se indutivamente: *a*) $\text{alf}(\text{stop}_U) = \text{alf}(x_U) = U$; *b*) $\text{alf}(!el.p) = \text{alf}(e)\cup\text{alf}(p)$; *c*) $\text{alf}(p + q) = \text{alf}(p)\cup\text{alf}(q)$, e *d*) $\text{alf}(\mu x.p) = \text{alf}(x) U \text{alf}(p)$.

Os subtermos definem-se *mutatis mutandis* como na secção anterior relativa à assinatura Σ_{QS}:

Definição 3.2 — A cada termo p em $\mathrm{Rec}_{TP}(X)$ associamos o conjunto dos seus *subtermos* definido indutivamente do modo seguinte: *a*) p é um subtermo de p; *b*) se p é $|e|.q$, então todo o subtermo de q é um subtermo de p; *c*) se p é p_1+p_2 ou $p_1 \| p_2$, então todo o subtermo de p_1 e todo o subtermo de p_2 é subtermo de p, e *c*) se p é $\mu x.p$, então todo o subtermo de q é um subtermo de p. O conjunto dos subtermos de p denota-se por sub(p).

Definição 3.3 — Um termo p de $\mathrm{Rec}_{TP}(X)$ diz-se *guardado* se em todo o subtermo recursivo $\mu x.q$ de p toda a ocorrência livre de x em q ocorre num subtermo da forma $|e|.r$ de q.

Definição 3.4 — Um termo de processo é um termo p em $\mathrm{Rec}_{TP}(X)$ que satisfaz as seguintes dependências de contexto: *a*) p é guardado; *b*) todo o subtermo $|e|.q$ de p satisfaz $\mathrm{alf}(e) \subseteq \mathrm{alf}(q)$; *c*) todo o subtermo $q+r$ de p satisfaz $\mathrm{alf}(q) = \mathrm{alf}(r)$, e *d*) todo o subtermo $\mu x.q$ de p satisfaz $\mathrm{alf}(x) = \mathrm{alf}(q)$.

Seja TProc(X) o conjunto de todos os termos de processo, fTProc(X) o conjunto dos termos de processo (sintacticamente) finitos e cTProc(X) o conjunto de todos os termos de processo fechados.

Definição/Proposição 3.5 — A álgebra-Σ_{TP} $TP = \langle |TP|, \leq_{TP}, \Sigma TP \rangle$ tem os seus elementos definidos como segue:

$$|TP| = \{\langle E,L\rangle : E \in \mathcal{P}_f(E) \text{ e } L \in \mathcal{P}(E^*)\}$$

$$\langle E,L\rangle \leq TP \langle E',L'\rangle \text{ se } E = E' \text{ e } L \subseteq L'$$

$$\Sigma_{TP} = \{\mathrm{stop}^{\cup}_{TP} : U \in \mathcal{P}_f(E)\} \cup \{|e|.TP : e \in E\} \cup \{+_{TP}, \|_{TP}\}$$

$$\begin{array}{rlll}
\mathrm{stop}^{\cup}_{TP} & : \rightarrow |QS| & \mathrm{stop}_{QS} & = \langle U, \{\varepsilon\}\rangle \\
|e|.\mathrm{TP} & : |TP| \rightarrow |TP| & |e|._{PS}(U,L\rangle) & = \langle \mathrm{alf}(e) \cup U, \{e\} \cup \{es : s \in L\}\rangle \\
+_{TP} & : |TP|\,|TP| \rightarrow |TP| & +_{TP}(\langle U,L\rangle, \langle U', L'\rangle) & = \langle U \cup U', L \cup L'\rangle \\
\|_{TP} & : |TP|\,|TP| \rightarrow |TP| & \|_{TP}(\langle U,L\rangle, \langle U', L'\rangle) & = \langle U \cup U', L \| L'\rangle
\end{array}$$

O sistema dos axiomas próprios *TP* apresenta-se no quadro II (o significado dos símbolos *R* e *T* será explicado adiante).

O teorema fundamental que pode ser derivado no sistema formal diz respeito a requisitos transaccionais: $\vdash_{TP} |ef|.x_U \leq |e|.|f|.x_U$. Este teorema encerra, nada mais nada menos, que o seguinte conteúdo: o processo denotado pelo termo $|ef|.x_U$ tem mais requisitos transaccionais que o processo denotado pelo termo $|e|.|f|.x_U$.

Proposição 3.6 — $\vdash_{TP} |ef|.x_U \leq |e|.|f|.x_U$ *TP7*

Prova:

$$
\begin{aligned}
&\vdash_{TP} |ef|.x_U \\
&\quad = |ef|.x_U + \mathrm{stop}_{U \cup \mathrm{alf}(e) \cup \mathrm{alf}(f)} \\
&\quad \leq |ef|.x_U + |e|.\,|f|.\,x_U \\
&\quad = |ef|.x_U + |e|.\,(|f|.\,x_U + \mathrm{stop}_{U \cup \mathrm{alf}(f)} \\
&\quad = |ef|.x_U + |e|.\,|f|.\,x_U + |e|.\,\mathrm{stop}_{U \cup \mathrm{alf}(f)} \\
&\quad = |ef|.x_U + |e|.\,\mathrm{stop}_{U \cup \mathrm{alf}(f)}) + |e|.\,|f|.\,x_U \\
&\quad = |e|.\,|f|.\,x_U + |e|.\,|f|.\,x_U \\
&\quad = |e|.\,|f|.\,x_U
\end{aligned}
$$

QUADRO II

Axiomatização TP

Estrutura

$$x_U + x_U = x_U \qquad TP_1$$

$$x_U + y_U = y_U + x_U \qquad TP_2$$

$$x_U + (y_U + z_U) = (x_U + y_U) + z_U \qquad TP_3$$

$$x_U + \mathrm{stop}_U = x_U \qquad TP_4$$

$$|e|.\,(x_U + y_U) = |e|.\,x_U + |e|.\,y_U \qquad TP_5$$

Reticulado

$$\mathrm{stop}_U \leq x_U \qquad TP_6$$

Transacções

$$|ef|.\,x_U + |e|.\,\mathrm{stop}_{U \cup \mathrm{alf}(f)} = |e|.\,|f|.\,x_U \qquad TP_8$$

Paralelismo

$$x_U \parallel y_V = R\,(Tx_U) \parallel (Ty_V) \quad \subseteq \qquad TP_9$$

A ordem parcial estrita relativa aos requisitos transaccionais introduz-se de modo análogo à ordem parcial estrita relativa aos requisitos de actividade:

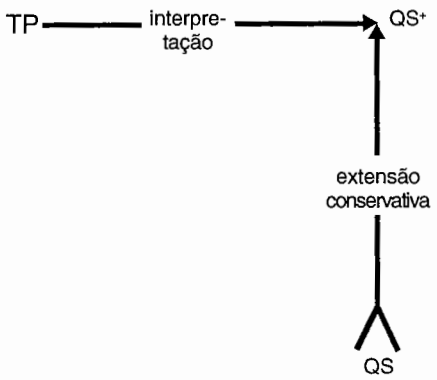
$$\langle E,L\rangle \leq_T \langle E',L'\rangle \text{ se } E=E',\ \mathrm{pc}(L) = \mathrm{pc}(L') \text{ e } L \quad L'$$

isto é, $\langle E,L\rangle \leq_T \langle E',L'\rangle$ se $\langle E,L\rangle \leq_{TP} \langle E',L'\rangle$ e $\mathrm{pc}(L) = \mathrm{pc}(L')$. Os requisitos transaccionais podem também ser investigados ignorando o axioma de reticulado e incluindo no sistema formal o teorema TP_7. Mais uma vez, a ordem parcial $\leq_{TP}$ resulta da sobreposição da ordem parcial estrita relativa aos requisitos transaccionais à ordem parcial comum relativa ao reticulado.

O resto desta secção será dedicado à implementação de termos de processo com requisitos transaccionais sobre uma extensão conservativa da teoria dos processos com requisitos de actividade.

FIGURA 3.1

Implementação de TProc(*X*) sobre Proc(*X*)



Seja $<\Sigma_{QS},QS>$ a apresentação da teoria inequacional de processos com requisitos de actividade, dada na secção 2, e $<\Sigma_{TP},TP>$ a apresentação da teoria acima de processos deterministas com requisitos transaccionais. Entendendo a assinatura Σ_{QS} com as operações de prefixação transaccional e juntando a *QS* as equações de tradução dadas no quadro III, providenciamos uma extensão conservativa de $<\Sigma_{QS},QS>$ que denotaremos por $<\Sigma_{QS^+},QS^+>$. Por conservativa queremos significar que não podemos derivar mais teoremas acerca da actividade (relativos a termos sobre Σ_{QS}) que não fossem já demonstráveis em *QS*.

QUADRO III

Tradução

$$|e|.\ |a.x_U = |ea|.\ x_U$$
$$a.x_U = |a|.\ x_U$$

As equações de tradução do quadro III permitem interpretar termos de processo com requisitos transaccionais através de termos de processo com requisitos de actividade (v. figura 3.1).

Por outro lado, $<\Sigma_{TP},TP>$ é equivalente a uma subapresentação de $<\Sigma_{QS^+},QS^+>$ (isto é, $\Sigma_{TP} \subseteq \Sigma_{QS^+}$ e $dc[(TP) \subseteq dc(QS^+)^*]$ onde as prefixações activa e passiva são símbolos de operação ocultos no sistema formal *TP*.

(*) Por *dc(X)* denotamos o conjunto de todos os teoremas que podem demonstrar-se no sistema *X*.

Podemos igualmente estender o domínio QS para um domínio QS^+ tal que QS^+ é gerado por $\Sigma_{QS^+}, QS+$ é correcto e adequado relativamente a QS^+ para termos de processo sobre Σ_{QS^+}, e tal que TP é uma sua restrição.

No que se segue (v. quadro IV) $\text{Proc}^+(X)$ é o conjunto de todos os termos de processo sobre Σ_{QS^+}.

Definição 3.7 — Seja $\mu: E^* \times \text{TProc}(X) \rightarrow \text{Proc}^+(X)$ uma aplicação que para cada par $<s,p>$ em $E^* \times \text{TProc}(X)$ retorna um termo misto em $\text{Proc}^+(X)$, definido indutivamente como se segue: *a*) para todo o $p \in \text{TProc}(X)$, $\mu(\varepsilon,p) = p$, e *b*) para todo o $e \in E$, para todo o $p \in \text{TProc}(X)$ e para todo o $s \in E^*$, $\mu(es,p) = !e.\mu(s,p)$.

Definição 3.8 — Seja $\sigma: E^* \times \text{TProc}(X) \rightarrow \text{Proc}^+(X)$ uma aplicação que para cada par $<s,p>$ em $E^* \times \text{TProc}(X)$ retorna um termo misto em $\text{Proc}^+(X)$, definido indutivamente como se segue: *a*) para todo o $p \in \text{TProc}(X)$, $\sigma(\varepsilon,p) = p$, e *b*) para todo o $e \in E$, para todo o $p \in \text{TProc}(X)$ e para todo o $s \in E^*$, $\sigma(es,p) = e.\mu(s,p)$.

A tradução pode ser efectuada indutivamente recorrendo às regras do quadro IV:

QUADRO IV

Tradução

$$
\begin{array}{rcl}
\text{stop}_U & \xrightarrow{T} & \text{stop}_U \\
x_U & \xrightarrow{T} & x_U \\
!e!.p & \xrightarrow{T} & \sigma(e, T(p)) \\
p + q & \xrightarrow{T} & T(p) + T(q) \\
\mu x.p & \xrightarrow{T} & \mu x.T(p)
\end{array}
$$

O cálculo de processos deterministas concorrentes com transacções pode ser logrado através de um procedimento sistemático em três etapas: *a*) uma tradução de $<\Sigma_{TP}, TP>$ para $<\Sigma_{QS}, QS>$ (em $<\Sigma_{QS^+} QS+>$); *b*) redução equacional do símbolo de composição empregando o sistema QS^+, e *c*) uma retradução (simbólica ou meta) para $<\Sigma_{TP}, TP>$. A retradução simbólica é intuitivamente definida apenas para os termos relevantes e apresentada no quadro V. A última regra é dependente do contexto (relembre que temos estado a usar símbolos do alfabeto latino para denotar transacções).

QUADRO V

Retradução

$$
\begin{array}{rcl}
\text{stop}_U & \xrightarrow{R} & \text{stop}_U \\
x_U & \xrightarrow{R} & x_U \\
a.p & \xrightarrow{R} & !a!.R(p)) \\
p + q & \xrightarrow{R} & R(p) + R(q) \\
\mu x.p & \xrightarrow{R} & \mu x.R(p) \\
 & & \\
!e!.R(!a.\ p) & \xrightarrow{R} & !ea!.R(p)
\end{array}
$$

Vamos agora exemplificar o cálculo: o termo de processo obtido por concorrência dos termos $|abc|.\ stop_{\{a,b,c\}}$ e $|a|.\ |e|.\ |f|.\ stop_{\{a,e,f\}}$ tem requisitos transaccionais *aebc* e *aebfc inter alia.* Omitindo os subtermos stop, a expansão completa é:

$$\vdash_{QS^+} |abc| \parallel |a|.\ |e|.\ |f|$$
$$= |abc|.\ |e|.\ |f|. + |abec|.\ |f| + |abefc| + |aebfc| + |aebc|.\ |f|. + |aebfc|$$

Outro exemplo, agora de *roll-back.* numa situação de impasse após envolvimento num primeiro evento comum:

$$\vdash_{QS^+} |ab|.\ stop_{\{a,b,d\}} \parallel |a|.\ |d|.\ stop_{\{a,b,d\}} = stop_{\{a,b,d\}}$$

A expansão completa é conduzida do modo seguinte:

$$\vdash_{QS^+} |ab|.\ stop_{\{a,b,d\}} \parallel |a|.\ |d|.\ stop_{\{a,b,d\}}$$

$$= a.\ !b.\ stop_{\{a,b,d\}} \parallel a.\ d.\ stop_{\{a,b,d\}}$$
$$= a.\ (!b.\ stop_{\{a,b,d\}} \parallel d.\ stop_{\{a,b,d\}})$$
$$= a.\ abort_{\{a,b,d\}}$$
$$= stop_{\{a,b,d\}}$$

Concluímos esta secção enunciando, sem provar, a correcção e adequação de *TP:*

Proposição 3.9 — Quaisquer que sejam os termos de processo p, $q \in cTProc(X)$, $\vdash_{TP} p \leq q$ se $\models_{TP} p \leq q$.

Prova: É essencialmente uma consequência de QS^+ ser redutivo relativamente a *QS*. Tecnicamente a prova segue os mesmos passos que a prova correspondente a *QS* após escolha de um conjunto conveniente de termos normais.

4 — Requisitos infinitários — modelo ωQS

Como vimos, a existência de termos de processo $abort_U$ reveste-se de especial significado no contexto da paralelização de requisitos transaccionais, constituindo uma ferramenta equacional para *roll-back.* Porém, $abort_U$ não só é o mais pequeno processo da ordem parcial completa QS_U, como acontece ser ponto fixo da equação $\lambda x_U.\ !e.\ x_U$ para todo o e em U, o que significa, entre outras coisas, que os termos de processo não passivamente guardados

degeneram em abort. Outro aspecto a salientar é que a composição de dois processos passivamente guardados pode ser redutível a abort:

$$\mu x.\ \text{moeda}.\ !\text{choc} \cdot x \parallel \mu x.\ !\text{moeda choc}.\ x = \text{abort}$$

Do ponto de vista da implementação de termos de processo com requisitos transaccionais nada temos a perder com este facto, já que as transacções são essencialmente finitárias.

Vamos apresentar um domínio, que permita capturar a semântica de sistemas com requisitos de actividade infinitários, incluindo a possibilidade de comportamento infinito. Nesta extensão do domínio semântico *QS* é possível trabalhar com termos de processo do tipo abort_U de modo a proceder a *rollback* e, ao mesmo tempo, podemos escolher um elemento minimal diferente de abort-run: o maior processo, capaz de realizar todas as possíveis computações sobre um determinado alfabeto. O novo domínio será denotado por ωQS.

Esta secção é dedicada aos processos ωQS, apresentando-se também um novo sistema formal. Voltamos a fixar um alfabeto contável *E*. Seja $A = E \cup \{\tau, \perp\}$, onde $\perp$ *denota* ***pausa*** *e* τ ***evento interno.***

Definição 4.1 — *A assinatura dos processos deterministas com requisitos infinitários é* $\Sigma_{\omega QS} = \Sigma_{\omega QS}{}^0 \cup \Sigma_{\omega QS}{}^1 \cup \Sigma_{\omega QS}{}^2$ onde $\Sigma_{\omega QS}{}^0 = \{\text{abort}_U, \text{stop}_U, \text{run}_U : U \in \mathcal{P}_f(E)\}$, $\Sigma_{\omega QS}1 = \{e. : e \in E\} \cup \{!e. : e \in E\}$ e $\Sigma_{\omega QS}{}^2 = \{+, \parallel\}$.

O conjunto dos termos recursivos sobre $\Sigma_{\omega QS}$ com variáveis em *X* denota-se por $\text{Rec}_{\omega QS}(X)$.

O alfabeto de um termo $p \in \text{Rec}_{\omega QS}(X)$ define-se indutivamente do modo seguinte: *a*) $\text{alf}(\text{abort}_U) = \text{alf}(\text{stop}_U) = \text{alf}(\text{run}_U) = \text{alf}(x_U) = U$; *b*) $\text{alf}(e.p) = \text{alf}(!e.p) = \{e\} \cup \text{alf}(p)$; *c*) $\text{alf}(p + q) = \text{alf}(p \parallel q) = \text{alf}(q)$, e *d*) $\text{alf}(\mu x.p) = \text{alf}(x) \cup \text{alf}(p)$.

Os subtermos definem-se *mutatis mutandis* como anteriormente.

Definição 4.2 — Um termo *p* de $\text{Rec}_{\omega QS}(X)$ diz-se *guardado* se em todo o subtermo recursivo $\mu x.q$ de *p* toda a ocorrência livre de *x* em *q*, ocorre num subtermo da forma *e.r* ou *!e.r* de *q*.

Definição 4.3 — Um termo de processo é um termo *p* em $\text{Rec}_{\omega QS}(X)$ que satisfaz as seguintes dependências de contexto: *a*) *p* é guardado; *b*) todo o subtermo *e.q* ou *!e.q* de *p* satisfaz $e \in \text{alf}(q)$; *c*) todo o subtermo $q + r$ de *p* satisfaz $\text{alf}(q) = \text{alf}(r)$, e *d*) todo o subtermo $\mu x.p$ de *p* satisfaz $\text{alf}(x) = \text{alf}(q)$.

Seja $\omega\text{Proc}(X)$ o conjunto de todos os termos de processo, $f\omega\text{Proc}(X)$ o conjunto dos termos de processo (sintacticamente) finitos e $c\omega\text{Proc}(X)$ o conjunto de todos os termos de processo fechados.

A seguir apresentamos o domínio semântico $\omega QS = < | \omega QS | , \leq_{\omega QS}, \Sigma_{\omega QS} >$. Mas, primeiro, temos de introduzir alguns conceitos topológicos.

Para todo o conjunto E, seja $E\omega$ o conjunto das sucessões de elementos de E. Uma sequência $<u_0, u_1, u_2, ...>$ de sucessões em E^ω converge para uma sequência u em E^ω se para todo $m \leq 0$ existe $n \geq 0$ tal que $u{\downarrow}m$ é $u_i{\downarrow}m$ para todo $i \geq n$. Neste caso definimos $\lim u_i = u$.

Seja u um elemento de E^ω e L um subconjunto de E^ω. Dizemos que u é um ponto de acumulação em L se existe uma sequência u_i em L tal que $\lim u_i = u$. O conjunto L diz-se fechado se L contém todos os seus pontos de acumulação. O fecho para limites de L, denotado por $\overline{L}$, consiste em todos os pontos de acumulação de L.

É trivial verificar que a classe de todos os conjuntos de sucessões fechados é fechada para uniões finitas e intersecções arbitrárias.

Retomemos o exemplo da máquina de vendas:

$$\text{MÁQUINA} = \mu x_{\{\text{moeda, choc}\}}.\ \text{moeda}.\ !\text{choc}.\ x_{\{\text{moeda, choc}\}}$$

cuja semântica pode, agora, ser dada em termos de sequências infinitas. Algumas sequências de comportamento possível são dadas pela expressão regular $(\perp^*\ \text{moeda}\ \perp^*\ \text{choc}\ \perp^*)^*\ \perp^\omega$. Esta expressão denota um conjunto aberto. Por exemplo, a sequência $\lambda n.$ *se* par(n) *então* moeda *se não* choc é comportamento limite também possível.

Definição/Proposição 4.4 — A álgebra $\omega QS = <|\omega QS|, \leq_{\omega QS}, \Sigma_{\omega QS}>$ tem os seus elementos definidos como segue:

$$|\omega QS| = \{<E,L>:\ E \in \mathcal{P}_f(E)\ \text{e}\ L \in \mathcal{P}(E^*)\}$$

$$<E,L> \leq_{QS} <E',L'> \text{ se } E = E' \text{ e } L \subseteq L'$$

$$\Sigma_{QS} = \{\text{abort}^{U}_{\omega QS},\ \text{stop}^{U}_{\omega QS},\ \text{run}^{U}_{\omega QS}:\ U \in \mathcal{P}_f(E)\} \cup \{e._{\omega QS}:\ !e \in E\} \cup \{+_{\omega QS},\ ||\omega_{QS}\}$$

$$\begin{array}{rll}
\text{abort}^{U}_{\omega QS} : \rightarrow |\omega QS| & & \text{abort}^{U}_{\omega QS} = <U,\varnothing> \\
\text{stop}^{U}_{\omega QS} : \rightarrow |\omega QS| & & \text{stop}^{U}_{\omega QS} = <U,\{\perp^\omega\}> \\
\text{run}^{U}_{\omega QS} : \rightarrow |\omega QS| & & \text{run}^{U}_{\omega QS} = <U,(U \cup \{\tau,\perp\})^\omega> \\
!e._{\omega QS} : |\omega QS| \rightarrow |\omega QS| & & !e._{\omega QS}(<U,L>) = <\{e\} \cup U, \perp^\omega; \{es:\ s \in L\}> \\
e._{\omega QS} : |\omega QS| \rightarrow |\omega QS| & & e._{\omega QS}(<U,L>) = !e._{\omega QS}(<U,L>) + \text{stop}^{U}_{\omega QS} \\
+_{\omega QS} : |\omega QS|\ |\omega QS| \rightarrow |\omega QS| & & +_{\omega QS}(<U,L>,<U',L'>) = <U \cup U', L \cup L'> \\
||_{\omega QS} : |\omega QS|\ |\omega QS| \rightarrow |\omega QS| & & ||_{\omega QS}(<U,L>,<U',L'>) = <U \cup U', L||L'>
\end{array}$$

$$L\ ||\ L' = \{\sigma \in (U \cup U' \cup \{\perp,\tau\})^\omega:\ \sigma{\downarrow}(U \cup \{\perp,\tau\}) \in L \text{ e } \sigma{\downarrow}(U \cup \{\perp,\tau\}) \in L'\} \cup \text{divergências}$$

onde *divergências* é o conjunto vazio excepto quando uma das componentes da composição é o processo run. Neste caso, se p não é $\text{abort}_{\text{alf}(p)}$, então $\text{run}_U\ ||\ p = \text{run}_{U \cup \text{alf}(p)}$. Tecnicamente, conseguimos a validade deste axioma

equacional à custa do elemento τ (daí que run já não é um processo determinista). Seja $\mathcal{U} = U \cup \{\bot,\tau\}$ e $\mathcal{U}' = U' \cup \{\bot,\tau\}$:

$$\text{divergências} = \{s;\sigma:\ s \in (\mathcal{U} \cup \mathcal{U}')^*,\ \sigma \in (\mathcal{U} \cup \mathcal{U}')^\omega \text{ e } (s{\downarrow}\mathcal{U};\tau^\omega \in L \text{ e } \exists \sigma' \in \mathcal{U}^\omega\ s{\downarrow}\mathcal{U}';\sigma' \in L') \text{ ou } (s{\downarrow}\mathcal{U}';\tau^\omega \in L' \text{ e } \exists \sigma' \in \mathcal{U}^\omega\ s{\downarrow}\mathcal{U};\sigma' \in L)\}$$

A ordem parcial de actividade estrita introduz-se do seguinte modo:

$$\langle E,L\rangle \leq_! \langle E',L'\rangle \text{ se } E = E',\ \mathrm{pc}(L) = \mathrm{pc}(L') \text{ e } L' \subseteq L$$

onde $\langle E,L\rangle \leq_! \langle E',L'\rangle$ se lê: $\langle E,L\rangle$ é menos activo que $\langle E',L'\rangle$. Note-se que a ordem parcial se encontra invertida quando comparada com $\leq_{QS}$ introduzida na secção 2.

Em ωQS a semântica do termo de processo RELÓGIO = $\mu x_{\{tic\}}$. !tic. $x_{\{tic\}}$ é $\{u \in \{\bot,tic\}^\omega:\ u{\downarrow}tic = tic^\omega\}$, como era desejado. De facto é a mais pequena solução da equação $!tic._{\omega QS}\ x_{\{tic\}} = x_{\eta tic\eta}$. Mas $\{u \in \{\bot,tic\}^\omega:\ u{\downarrow}tic = tic^\omega\}$ não é solução da equação $tic._{\omega QS}\ x_{\{tic\}\eta tic\eta}$; a mais pequena solução é $\{\bot,tic\}^\omega$.

Note-se que o fecho topológico corresponde ao fecho para prefixos. De facto, se as sequências de $(\bot^*$ moeda $\bot^*$ choc $\bot^*)^*\ \bot^\omega$ se contam entre os traços quiescentes da máquina, então no fecho também temos as sequências:

$$\begin{aligned} u_0 &= \text{moeda choc } \bot^\omega \\ u_1 &= \text{moeda } \bot\text{choc } \bot^\omega \\ u_2 &= \text{moeda } \bot\bot\text{choc } \bot^\omega \\ u_3 &= \text{moeda } \bot\bot\bot\text{choc } \bot^\omega \\ &\cdots \\ u_n &= \text{moeda } \bot^n \text{ choc } \bot^\omega \\ &\cdots \\ \lim u &= \text{moeda } \bot^\omega \end{aligned}$$

Proposição 4.5 — O terno $< |\ \omega QS\ |\ ,\leq_{\omega QS},\Sigma_{\omega QS}>$ é um domínio-$\Sigma_{\omega QS}$, isto é, $\langle \omega QS_U,\leq_{\omega QS}\rangle$ é uma ordem parcial completa e todas as funções em $\Sigma_{\omega QS}$ são contínuas.

Prova: Para cada U, ωQS_U está parcialmente ordenado por $\leq_{\omega QS}$; o elemento inicial é o processo $\mathrm{run}^U_{\omega QS}$. *a*) $\langle \omega QS_U,\leq_{\omega QS}\rangle$ é uma ordem parcial completa: seja C um conjunto orientado de processos em ωQS_U, o limite de C é o processo, $\lim_{\omega QS} C = \langle U,\cap_{P \in C} \tau(P)\rangle$. *b*) Não oferece dificuldade provar a continuidade das funções em $\Sigma_{\omega QS}$.

Listamos no quadro VI a axiomatização de ωQS na suposição de que $a \neq b$ sempre que a e b surgem no mesmo axioma-esquema.

A correcção de cada um dos axiomas específicos é rotineira, isto é, se $\vdash_{\omega QS} p \leq q$ então $\models_{\omega QS} p \leq q$, quaisquer que sejam os termos de processo p e q. O combinador || é igualmente redutivo. Porém, o domínio semântico já não é gerado (sobrejectividade). Só uma extensão da sintaxe de modo a contemplar o não determinismo (temos já as divergências) possibilitará recuperar a sobrejectividade. Desconhecemos ainda se o sistema formal é adequado relativamente a ωQS, para termos em $c\omega Proc(X)$, embora conjecturemos que sim.

Conjectura 4.6 — Quaisquer que sejam os termos p e q em $cf\omega Proc(X)$, se $\models_{\omega QS} p \leq q$, então $\vdash_{\omega QS} p \leq q$.

QUADRO VI

Axiomatização QS

Estrutura

$x_U + x_U$	=	x_U	ωQS_1
$x_U + y_U$	=	$y_U + x_U$	ωQS_2
$x_U + (y_U + z_U)$	=	$(x_U + y_U) + z_U$	ωQS_3
$x_U + \text{abort}_U$	=	x_U	ωQS_4
$x_U + \text{run}_U$	=	run_U	ωQS_5
$!a.(x_U + y_U)$	=	$!a.x_U + !a.y_U$	ωQS_6

Reticulado

run_U	$\leq$	$x_U \leq \text{abort}_U$	ωQS_7

Roll-back

$!a.\text{abort}_U$	=	$\text{abort}_{U \cup \{a\}}$	ωQS_8

Actividade

$!a.x_U + \text{stop}_{U \cup \{e\}}$	=	$a.x_U$	ωQS_9

Paralelismo

$(y_V + z_V) \,\|\|\, x_U$	=	$y_V \,\|\|\, x_U + z_V \,\|\|\, x_U$	ωQS_{10}
$x_U \,\|\|\, y_V$	=	$y_V \,\|\|\, x_U$	ωQS_{11}
$x_U \,\|\|\, \text{abort}_V$	=	$\text{abort}_{U \cup V}$	ωQS_{12}
$x_U \,\|\|\, \text{run}_V$	=	$\text{run}_{U \cup V}$ se $x \neq \text{abort}$	ωQS_{13}
$!a.x_U \,\|\|\, \text{stop}_{V \setminus \{a\}}$	=	$!a.(x_U \,\|\|\, \text{stop}_{V \setminus \{a\}})$	ωQS_{14}
$!a.x_U \,\|\|\, \text{stop}_{V \cup \{a\}}$	=	$\text{abort}_{U \cup V \cup \{a\}}$	ωQS_{15}
$!a.x_{U \setminus \{b\}} \,\|\|\, !b.\, y_{V \setminus \{a\}}$	=	$!a.(x_{U \setminus \{b\}} \,\|\|\, !b.y_{V \setminus \{a\}})$	
	+	$!b.\, (!a.x_{U \setminus \{b\}} \,\|\|\, y_{V \setminus \{a\}})$	ωQS_{16}
$!a.x_{U \cup \{b\}} \,\|\|\, !b.y_{V \cup \{a\}}$	=	$\text{abort}_{U \cup V \cup \{a,b\}}$	ωQS_{17}
$!a.x_{U \cup \{b\}} \,\|\|\, !b.y_{V \setminus \{a\}}$	=	$!a.(x_{U \cup \{b\}} \,\|\|\, !b.y_{V \setminus \{a\}})$	ωQS_{18}
$!a.x_U \,\|\|\, !a.y_V$	=	$!a.(x_U \,\|\|\, y_V)$	ωQS_{19}

5 — Conclusões

Introduzimos um modelo de processos sequenciais com requisitos de animação no paradigma da intercalação. Os requisitos de animação explicam por que razão um processo se envolve expontaneamente nalgumas acções e noutras não. A ideia básica consiste em remover da semântica de traços tradicional a condição de fecho para prefixos: um traço s pertence à semântica Λ de um dado processo se o estado após a execução de s é quiescente, isto é, é um estado no qual o processo não está compelido à execução de nenhuma acção. A composição paralela de requisitos de animação foi caracterizada através de uma axiomatização correcta e adequada e reflecte o facto de que não é possível intercalar um evento não expontâneo entre dois eventos expontâneos contíguos sem primeiramente o activar.

Em seguida, aplicámos esta teoria ao estudo de processos com requisitos transaccionais. Para este fim, introduzimos uma linguagem conveniente de termos de processo com transacções e um mecanismo de tradução desta linguagem para a linguagem dos termos de processo com requisitos de animação. Reciprocamente, foi possível mostrar que quase todos os processos com requisitos de animação têm retradução para termos de processo com requisitos transaccionais. Através de uma técnica de tradução/retradução, aplicámos a teoria (in)equacional já usada para compor termos de processo com requisitos de animação à composição de termos de processo com transacções, obtendo assim uma teoria de sincronização de transacções.

Porém, não é fácil capturar no modelo proposto processos com requisitos de animação infinitos, tal como no relógio denotado por:

$$\text{RELÓGIO} = \mu x_{\{\text{tic}\}}.\ !\text{tic}.\ x_{\{\text{tic}\}}$$

porque não há sequer um traço em {tic}* após o qual o RELÓGIO se encontra num estado quiescente. Assim, termos de processo recursivos passivamente guardados podem ser representados por conjuntos infinitos de traços. Mas os termos de processo recursivos activamente guardados não podem ser representados deste modo. O RELÓGIO tem de executar uma «transacção» infinita que não pode ser cometida; deste modo, o conjunto dos traços quiescentes é vazio. De modo a capturar os cometimentos infinitos juntámos sequências infinitas (de modo a que o RELÓGIO com o cometimento recursivo tic é o processo $\langle\{\text{tic}\},\{\text{tic}^{\omega}\}\rangle$).

A extensão das assinaturas será alvo de investigação futura. Por exemplo, para todo o processo p em Proc(X) seja $\{a\}p$ um novo termo de processo que denota o processo em que a é um requisito de animação de p (que reflecte o operador modal F da lógica temporal). Este tipo de extensões da te-

oria equacional dos processos com requisitos de animação será o ponto de partida para estabelecer a ponte entre a lógica temporal e a lógica equacional com recursividade. Pensamos investigar um esquema de transformação de uma lógica para a outra de modo a providenciar um mecanismo de construção de termos de processo a partir de especificações temporais por aplicação do «princípio da programação transformacional» advogado pelo Projecto de Munique CIP (Bauer *et al,* 1985; Bauer *et al,* 1987) e no livro de Olderog (Olderog, 1991).

REFERÊNCIAS

(Bauer et al, 1985), F. L. Bauer *et al,* «The Munich Project CIP, vol. I: The Wide Spectrum Language CIP-L», in *Lecture Notes in Computer Science,* 183, Springer Verlag, 1985.

(Bauer et al, 1987), F. L. Bauer *et al,* «The Munich Project CIP, vol. II: The Program Transformation System CIP-S», in *Lecture Notes in Computer Science,* 292, Springer Verlag, 1987.

(Bernstein *et al,* 1987), P. A. Bernstein, V. Hadzilacos e N. Goodman, *Concurrency Control and Recovery in Database Systems,* Addison Wesley, 1987.

(Brookes *et al,* 1984), S. Brookes, C. A. R. Hoare e A. Roscoe, «A Theory of Communicating Sequential Processes», in *Journal of the ACM,* 31, 7, 1984.

(Cellary *et al,* 1988), W. Cellary, E. Gelenbe e T. Morzy, *Concurrency Control in Distributed Database Systems,* North Holland, 1988.

(Costa, 1989), J. F. Costa, *Teoria Algébrica dos Processos Animados,* tese de mestrado, Universidade Técnica de Lisboa, Setembro de 1989.

(Gifford e Donahue, 1985), D. K. Gifford e J. E. Donahue, «Coordinating Independent Atomic Actions», in *Proceedings COMPCON 85,* 1985.

(Gorrieri e Montanari, 1990), R. Gorrieri e U. Montanari, «SCONE: A Simple Calculus of Nets», in *Proceedings of Concur'90 — Theories of Concurrency: Unification and Extension,* Springer Verlag, 1990, 2-30.

(Gray, 1980), J. Gray, «A Transactional Model», in *Lecture Notes in Computer Science,* 85, Springer Verlag, 1980.

(Hennessy, 1988), M. Hennessy, *Algebraic Theory of Processes,* MIT Press, 1988.

(Hoare, 1985), C. A. R. Hoare, *Communicating Sequential Processes,* Prentice Hall, 1985.

(Jonsson, 1985), B. Jonsson, «A Model and Proof System for Asynchronous Networks», in *Proceedings of the 4th Annual ACM Symposium on Principles on Distributed Computing,* Minaki, Canada, 1985.

(Kanellakis, 1981), P. C. Kanellakis, *The Complexity of Concurrency Control for Distributed Databases,* tese de doutoramento, MIT, 1981.

(Misra e Chandy, 1981), J. Misra e K. M. Chandy, «Proofs of networks of processes», in *IEEE trans. Software Eng.,* 7, 1981, 417-426.

(Misra *et al,* 1982), J. Misra, K. Chandy e T. Smith, «Proving safety and liveness of communicating processes with examples», *in Proceedings ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* 1982, 201-208.

(Misra, 1984), J. Misra, «Reasoning About Networks of Communicating Processes», in *INRIA Advanced Nato Study Institute on Logics and Models for Verification and Specification of Concurrent Systems,* Nice, France, 1984.

(Nielsen *et al,* 1989), M. Nielsen, U. Engberg e K. Larsen, «Fully Abstract Models for a Process Language with Refinement», in *Lecture Notes in Computer Science,* 354, Springer Verlag, 1989.

(Olderog e Hoare, 1986), E. R. Olderog e C. A. R. Hoare, Specification-oriented Semantics for Communicating Processes», in *Acta Informatica,* 23, 1986, 9-66.

(Olderog, 1991), E. R. Olderog, «Nets Terms and Formulas», *Cambridge Tracts in Theoretical Computer Science,* 23, Cambridge University Press, 1991.

(Van Snepscheut, 1985), J. Van de Snepscheut, «Trace Theory and VLSI Design», in *Lecture Notes in Computer Science,* 200, Springer Verlag, 1985.