

Universidade do Minho

2005/2006		1.º Semestre	2.º Semestre	Anual
		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DISCIPLINAS	Métodos Formais de Programação II (7008N2) + Opção II — Métodos Formais de Programação II (5308P3)	DOCENTE J.N. Oliveira – 406006		
CURSOS	LMCC + LESI			

AULA	SUMÁRIO
<p>Teórica 2006.02.20 2.ª-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p>Apresentação da disciplina. Equipa docente. Programa da disciplina e seu enquadramento no plano de estudos. Regime de avaliação. Trabalho opcional. Bibliografia. Informação electrónica sobre a disciplina: www.di.uminho.pt/~jno/html/mii.html. <i>Introdução</i>: A “equação de Wirth” [9]</p> <p style="text-align: center;"><i>Algorithms + Data Structures = Programs</i></p> <p>e o binómio <i>especificação / implementação</i>. Necessidade de eficiência. Eficiência das implementações <i>versus</i> clareza das especificações. Noção de <i>detalhe</i> de implementação. Múltiplas implementações para a mesma especificação (modelo). Necessidade de estratégias (ordens) para o refinamento.</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.02.27 2.ª-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Não houve aula (aula a compensar logo que possível).</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.02.27 2.ª-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p>Não houve aula (aula a compensar logo que possível).</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.03.06 2.ª-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Revisões de <i>Métodos Formais de Programação I</i>: prática em modelação VDM:</p> <ul style="list-style-type: none"> •Caso de estudo ECTSs — formulação do invariante do modelo. <p style="text-align: right;">(v.s.f.f.)</p>

(cont.)	<p>Exercício 1. Demonstrar a implicação</p> $f \cdot r = id \Rightarrow f \text{ é sobrejectiva e } r \text{ é injectiva} \quad (1)$ <p><i>Sugestão:</i> começar por $r \subseteq f^\circ$</p> <p>□</p> <p>Exercício 2. [Transformada-PF]</p> <ul style="list-style-type: none"> • Converter para variáveis e interpretar as expressões $f \cdot \leq \subseteq \sqsubseteq \cdot g \quad (2)$ $f^\circ \cdot \leq = \sqsubseteq \cdot g \quad (3)$ <p>onde \leq e \sqsubseteq são ordens. <i>Sugestão:</i> aplique a <i>regra do guardanapo</i> (214) sempre que possível.</p> <ul style="list-style-type: none"> • Mostrar que se se fizer $\leq := id$ em (2), esta expressão traduz a ordem <i>ponto-a-ponto</i> entre funções (216). <p>□</p> <p>O DOCENTE _____</p>
---------	---

AULA	SUMÁRIO
<p>Teórica 2006.03.06 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Introdução ao refinamento de modelos:</i> Princípios gerais de substituição de um modelo por outro.</p> <p>Nível algorítmico: relação de satisfação (substituição, ou refinamento). A tradicional ordem <i>s menos definido do que r</i> entre autómatos de estados finitos (vistos como funções de estado actual para estados sucessores):</p> $s \vdash r \stackrel{\text{def}}{=} \langle \forall a : (s \ a) \supset \emptyset : \emptyset \subset (r \ a) \subseteq (s \ a) \rangle \quad (4)$ <p>Nível dos dados: necessidade de mudança de formato dos dados. Caso mais simples: formatos isomorfos. Relaxe da relação de isomorfismo $A \cong B$ à relação $A \leq B$ captando o facto de A ter menos informação que B. Relações de abstracção (simples + sobrejectivas) e de representação (injectivas e inteiras). Assim: isomorfismo f° (converso de f), ambos tais que $\text{img } f = id$ e $\text{ker } f = id$ é relaxado a r tal que $f \cdot r = id$, deixando de ser exigido $r \cdot f = id$.</p> <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.03.13 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Conclusão dos exercícios da aula anterior. Prática em modelação VDM:</p> <ul style="list-style-type: none"> • Caso de estudo HTable — modelo de tabelas de “hashing” (variante <i>listas de colisões</i>) • Htable vista como refinamento de Collection — funções de abstracção e de representação. <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.03.13 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Refinamento formal de dados</i> : lei da representação por “apontador”:</p> $A \leq A + 1 \quad (5)$ <p>Estudo do isomorfismo de totalização de correspondências:</p> $A \multimap B \cong (B + 1)^A \quad (6)$ <p>Início do estudo do repertório de <i>inequações de refinamento</i> e respectivas funções de abstracção e de representação:</p> $\mathcal{P}A \leq A^* \quad (7)$ $\mathcal{P}A \leq \mathbf{N} \multimap A \quad (8)$ $\mathcal{P}A \leq A \multimap B \quad (9)$ $\mathcal{P}A \leq A \multimap \mathbf{N} \quad (10)$ <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.03.20 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Conclusão de alguns exercícios pendentes. Refinamento: cálculo do inverso do isomorfismo <i>split</i> associado à lei</p> $(B \times C)^A \cong B^A \times C^A \quad (11)$ <p>Prática em modelação VDM:</p> <ul style="list-style-type: none"> •Caso de estudo ECTSs — formulação dos invariantes de <i>integridade referencial</i> e de duas funcionalidades. <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.03.20 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Refinamento de dados por cálculo (cont.)</i> : Estudo do repertório de <i>inequações de refinamento</i> e respectivas funções de abstracção e de representação:</p> $A \multimap B \times C \leq (A \multimap B) \times (A \multimap C) \quad (12)$ $(B + C) \multimap A \cong (B \multimap A) \times (C \multimap A) \quad (13)$ $A \multimap (B + C) \leq (A \multimap B) \times (A \multimap C) \quad (14)$ $(A \multimap B)^C \cong (C \times A) \multimap B \quad (15)$ <p>A projecção relacional (238) e seu caso particular para relações simples $f \multimap g$ que é usado em funções de abstracção/representação.</p> <p style="text-align: right;">(v.s.f.f.)</p>

(cont.)	<p>Apresentação da lei</p> $A \rightarrow D \times (B \rightarrow C) \leq (A \rightarrow D) \times ((A \times B) \rightarrow C) \quad (16)$ <p style="text-align: right;">O DOCENTE _____</p>
---------	---

AULA	SUMÁRIO
<p>Prática 2006.03.27 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Cálculo da simplicidade de uma projecção $\pi_{g,f}R$ (238) e sua relação com dependências funcionais em bases de dados. Projecções de relações simples. A notação $f \rightarrow g = \pi_{g,f}$ para f injectiva. Introdução ao VDM++. Exemplo: os modelos <code>stackAlg.vpp</code> e <code>stackObj.vpp</code>.</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.03.27 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Refinamento de dados por cálculo</i> : Propriedades da relação \leq: reflexividade e transitividade e suas provas. Relacionadores (<i>relators</i>) e o refinamento estruturado. Análise detalhada das representações e abstracções das leis estudadas até ao momento.</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.04.03 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Apresentação do tema proposto para o projecto da disciplina. Exercícios de refinamento relacional dos modelos <i>BAMS</i> e <i>ECTS</i> de <i>Métodos Formais de Programação I</i>.</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.04.03 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Refinamento formal de dados (cont.)</i> : Introdução ao refinamento de modelos de dados recursivos. Necessidade de <i>hilomorfismos</i> relacionais para exprimir abstrações e representações. O hilomorfismo $\llbracket R, S \rrbracket$ como solução da equação relacional</p> $X = R \cdot F X \cdot S$ <p>Exemplos introdutórios: “fold” de conjuntos e de <i>mappings</i>. <i>Introdução ao cálculo de pontos-fixos</i> : Funções monótonas, pré/pós-pontos-fixos. Teorema de Tarski. Notação μ. Resolução de equações relacionais. Casos típicos: hilo-equações $X = R \cdot \underbrace{(F X)}_{f X} \cdot S$ e outras, por exemplo,</p> $X = R \cup \underbrace{R \cdot X}_{g X}$ <p>(cf. fecho transitivo.)</p> <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.04.10 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Demonstração das propriedades “functoriais” da projecção relacional</p> $id \rightarrow id = id \quad (17)$ $id \rightarrow (g \cdot h) = (id \rightarrow g) \cdot (id \rightarrow h) \quad (18)$ <p>Continuação dos exercícios de refinamento relacional da aula anterior. Conversão do modelo ECTS para diagrama entidades-relações. Exercícios de especificação recursiva sobre modelos com funções parciais finitas. O caso de estudo <i>FS</i> (“file system”).</p> <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.04.10 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p>Teorema da <i> fusão-μ</i>: para h, g monótonas e f^b adjunta-inferior,</p> $f^b(\mu h) = \mu g \iff f^b \cdot h = g \cdot f^b \quad (19)$ (20) <p>Aplicações: prova de</p> $\llbracket S, R \rrbracket^\circ = \llbracket R^\circ, S^\circ \rrbracket$ <p>e das leis de fusão-hilo:</p> <p style="text-align: right;">(v.s.f.f.)</p>

(cont.)	$V \cdot \llbracket S, R \rrbracket = \llbracket T, R \rrbracket \iff V \cdot S = T \cdot (FV)$ $\llbracket S, R \rrbracket \cdot V = \llbracket S, U \rrbracket \iff R \cdot V = FV \cdot U$ <p>Os tipos colectivos de dados em VDM-SL e seus hilomorfismos. O tipo $\text{map } A \text{ to } B$. O tipo $\text{set of } A$ e a notação $\{\{g\}\}$. Catamorfismos relacionais. Predicados indutivos vistos como catamorfismos coreflexivos.</p> <p style="text-align: right;">O DOCENTE _____</p>
---------	---

AULA	SUMÁRIO
Prática 2006.04.24 2. ^a -feira, 11h00–13h00 DI-0.04 (LMCC+LESI)	<p>Não houve aula (aula a compensar logo que possível).</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
Teórica 2006.04.24 2. ^a -feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)	<p>Não houve aula (aula a compensar logo que possível).</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
Prática 2006.05.08 2. ^a -feira, 11h00–13h00 DI-0.04 (LMCC+LESI)	<p>Resolução dos exercícios seguintes, onde as funções, relações e tipos de dados do módulo <code>mi i 0506.vdm</code> que se referem podem ser procuradas consultando o anexo <i>Function/Method Cross-Reference Index</i> que é gerado automaticamente:</p> <p>Exercício 3. Procure a função <i>puts</i> e as relações <i>Puts</i> e <i>Listify'</i>.</p> <ol style="list-style-type: none"> Mostre que <i>Puts</i> é simples. Desenhe o diagrama do anamorfismo relacional $Listify' = \llbracket Ins^\circ \rrbracket$, onde $Ins \stackrel{\text{def}}{=} [\emptyset, Puts]$. Mostre — recorrendo às leis (240) a (256) — que $Listify' \subseteq elems^\circ \tag{21}$ <p>onde <i>elems</i> (primitiva em VDM-SL) é o catamorfismo $\llbracket [\emptyset, puts] \rrbracket$.</p> De (21) infere-se $elems \cdot Listify' \subseteq id$. Contudo, $Listify' \cdot elems \subseteq id$ não se verifica. Porquê? Apresente um contra-exemplo. <p style="text-align: right;">□ (v.s.f.f.)</p>

<i>(cont.)</i>	<p>Exercício 4. Especifique em notação VDM-SL a função (recursiva) <i>tar</i>. Mostre que <i>tar</i> não é um isomorfismo. \square</p> <p>Exercício 5. [Transformada-PF] Mostre, usando (235), que a reflexão-+ (236) de um coproduto $A + B$ é equivalente a</p> $\text{img } i_1 \cup \text{img } i_2 = id \quad (22)$ <p>por sua vez equivalente a</p> $\left\langle \begin{array}{c} \forall x : \\ x \in A + B : \\ \langle \exists a : a \in A : x = i_1 a \rangle \\ \vee \\ \langle \exists b : b \in B : x = i_2 b \rangle \end{array} \right\rangle \quad (23)$ <p>o que capta bem a ideia da união disjunta de A e B retida em $A + B$. \square</p> <p>O DOCENTE _____</p>
----------------	---

AULA	SUMÁRIO
<p>Teórica 06.05.08 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p>Polimorfismo paramétrico — porquê e para quê? Teorema “grátis” de um tipo polimórfico t. Operador de Reynolds:</p> $f(R \leftarrow S)g \equiv f \cdot S \subseteq R \cdot g$ <div style="text-align: center;"> </div> <p>Exemplos. Teorema “grátis” do operador $\langle _ \rangle$</p> $f \cdot B \langle R, S \rangle \subseteq S \cdot g \Rightarrow \langle f \rangle \cdot F R \subseteq S \cdot \langle g \rangle$ <p>e seus corolários. Leis de fusão e de absorção.</p> <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.05.15 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Não houve aula (tolerância do Enterro da Gata).</p> <p>O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 2006.05.15 2.^a feira, 14h00-16h00 (Aula suplementar)</p>	<p><i>Refinamento formal de dados (cont.)</i> : Teorema de desrecursivação genérica:</p> $\mu F \leq (K \rightarrow F K) \times K \quad (24)$ <p style="text-align: right;"><i>(v.s.f.f.)</i></p>

(cont.)	<p>Exemplos: desrecursivação do tipo de dados $Exp \cong A + B \times Exp^*$. Primeira análise da desrecursivação do modelo</p> $ \begin{array}{ll} GenDia :: & indiv : token \quad /*data about an individual */ \\ & mother : [GenDia] \\ & father : [GenDia] \end{array} $ <p>Estudo da função de abstracção de abstracção associada a (24):</p> $\sim f(M, k) \stackrel{\text{def}}{=} \llbracket M \rrbracket k \quad (25)$ <p>Início do estudo do respectivo invariante concreto. Relações de pertença estrutural (\in_F) e acessibilidade estrutural (\prec_σ).</p> <p style="text-align: right;">O DOCENTE _____</p>
---------	--

AULA	SUMÁRIO
<p>Prática 2006.05.22 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Resolução de exercícios sobre <i>teoremas grátis</i> e desrecursivação de modelos:</p> <p>Exercício 6. No contexto do processo de desrecursivação do modelo <i>GenDia</i> da aula anterior, calcule a função de abstracção associada ao passo de desrecursivação (exprimindo-a em notação VDM-SL). \square</p> <p>Exercício 7. Após calcular o teorema grátis do tipo de</p> $ \begin{array}{l} sort : a^* \leftarrow a^* \leftarrow (2 \leftarrow (a \times a)) \\ nub : a^* \leftarrow a^* \leftarrow (2 \leftarrow (a \times a)) \end{array} $ <p>investigue a sua instanciação para</p> <ul style="list-style-type: none"> • <i>sort</i> p em que p define uma ordem total e • <i>nub</i> p em que p define uma equivalência <p>\square</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Teórica 06.05.22 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Técnicas de refinamento algorítmico</i> : Versão-PF da definição (4):</p> $S \vdash R \equiv (\text{dom } S \subseteq \text{dom } R) \wedge (R \cdot \text{dom } S \subseteq S)$ <p>A <i>eficiência</i> como principal motivação para o refinamento algorítmico. Refinamento funcional — satisfação de uma especificação implícita S por uma função f:</p> $S \vdash f \equiv f \cdot \text{dom } S \subseteq S \quad (26)$ <p>Exemplo: resolução da equação</p> $IsPermutation \vdash f$ <p style="text-align: right;">(v.s.f.f.)</p>

<i>(cont.)</i>	<p>em ordem a f, sabendo que $IsPermutation = \ker seq2bag$, onde</p> $seq2bag = ([[bnil, bcons]]) \quad (27)$ $bnil = \{\mapsto\} \quad (28)$ $bcons = \oplus \cdot (singb \times id) \quad (29)$ $singb\ a = \{a \mapsto 1\} \quad (30)$ <p style="text-align: right;">O DOCENTE _____</p>
----------------	--

AULA	SUMÁRIO
<p>Prática 2006.05.29 2.^a-feira, 11h00–13h00 DI-0.04 (LMCC+LESI)</p>	<p>Resolução dos exercícios seguintes:</p> <p>Exercício 8. No contexto do processo de desrecursivação do modelo <i>GenDia</i> da aula anterior, calcule</p> <ul style="list-style-type: none"> •A relação de pertença associada ao relator do respectivo tipo de dados $\in_F = i_2^\circ \cdot \pi_2 \cup i_2^\circ \cdot \pi_3 \quad (31)$ <ul style="list-style-type: none"> •A respectiva relação de acessibilidade $\prec_\sigma = \in_F \cdot \sigma \quad (32)$ <p>□</p> <p>Exercício 9. Completar o cálculo que se segue:</p> $ \begin{aligned} & S \vdash f \\ \equiv & \{ \text{definição (26)} \} \\ & \dots \\ \equiv & \{ \text{Galois} \} \\ & \dots \\ \equiv & \{ \dots \} \\ & y(\text{dom } S)x \Rightarrow y(f^\circ \cdot S)x \\ \equiv & \{ \dots \} \\ & y = x \wedge x \in \text{dom } S \Rightarrow (f\ y)\ S\ x \\ \equiv & \{ \dots \} \\ & x \in \text{dom } S \Rightarrow (f\ x)\ S\ x \end{aligned} $ <p>□</p> <p style="text-align: right;">(v.s.f.f.)</p>

(cont.)	<p>Exercício 10. Aplicar o resultado do exercício anterior à verificação do facto $S \vdash id$ onde S é a especificação que se segue, escrita em VDM-SL:</p> <pre> S(n: real) r: real pre n > 1 post r*r + 2*n*n = 3*n*r; </pre> <p>Será id a única implementação funcional de S? Justificar informalmente. \square</p> <p>Exercício 11. Demonstrar as seguintes propriedades da relação \vdash:</p> $\perp \vdash f \quad , \quad \top \vdash f \quad (33)$ $(S \cup R) \vdash f \iff S \vdash f \wedge R \vdash f \quad (34)$ $(S \cap R) \vdash f \iff S \vdash f \wedge R \vdash f \quad (35)$ $(\ker g) \vdash f \iff g \cdot f = g \quad (36)$ $g \vdash f \iff f = g \quad (37)$ <p>\square</p> <p style="text-align: right;">O DOCENTE _____</p>
---------	---

AULA	SUMÁRIO
<p>Teórica 06.05.29 2.^a-feira, 14h00–16h00 Sala DI-A1 (LESI+LMCC)</p>	<p><i>Técnicas de refinamento algorítmico (conclusão)</i> : Leis do refinamento relacional — satisfação progressiva de uma especificação implícita S por outra especificação R:</p> $S \vdash R \iff R \cdot \text{dom } S \subseteq S \wedge \text{dom } S \subseteq \text{dom } R \quad (38)$ <p>Propriedades de ordem parcial da relação \vdash. Propriedade de F-monotonia. O refinamento funcional $f \vdash g$ e a procura de eficiência. Leis de fusão “vertical” de processos algorítmicos (leis functoriais, de fusão e de absorção). Leis de fusão “horizontal” de processos algorítmicos. A lei de Fokkinga (257) e o seu corolários “banana-split” (258). Cálculo de ciclos <code>for/while</code> (250) a partir do hilomorfismo genérico (251)</p> $f : A \longrightarrow C$ $f = p \rightarrow b, \theta \cdot \langle d, f \cdot e \rangle$ <p>Introdução de parâmetros de acumulação. Leis de factorização iterativa (252, 253). Uso do cálculo de hilomorfismos na prova de (252). Considerações finais. Encerramento da disciplina. Preenchimento dos inquéritos de avaliação das aulas teóricas e práticas.</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
Prática 2006.05.31 4. ^a feira, 14h00-16h00 (Aula suplementar)	<p>Resolução dos exercícios seguintes relativos à factorização iterativa de hilomorfismos:</p> <p>Exercício 12. Identifique que lei (252) ou (253) foi aplicada para se obter, em VDM-SL, os pares de funções que se seguem e reconstitua o esquema linear de que se partiu em cada caso:</p> <pre> h(y) == haux(1, y); haux(b, y) == if y = [] then b else haux(b * hd y , tl y); g(f, y) == gaux(f, {}, y); gaux(f, b, y) == if y = {} then b else let x in set y in gaux(f, {f(x)} union b , y \ {x}); f(y) == faux(1, y); faux(b, y) == if y = 0 then b else faux(y * b , y -1); j(x, y) == jaux(x, false, y); jaux(x, b, y) == if y = [] then b else jaux(x, (x = hd y) or b , tl y); </pre> <p>Isto é, para cada caso identifique os parâmetros p, d, e etc em (251). \square</p> <p>Exercício 13. É possível mostrar que o cálculo da projecção $g \cdot M \cdot f^\circ$, em que M é um <i>mapping</i> finito e f é injectiva, se pode fazer em VDM-SL através do hilomorfismo</p> <pre> proj(g)(f)(M) == if M = { -> } then { -> } else let a in set dom M in {f(a) -> g(M(a))} munion proj(g)(f)({a} <-: M) </pre> <p>Mostre que é possível calcular a mesma projecção de forma iterativa, isto é, sob a forma de um ciclo-while. \square</p> <p style="text-align: right;">O DOCENTE _____</p>

AULA	SUMÁRIO
<p>Prática 2006.06.20 3.^a feira, 10h00-13h00 (Aula suplementar)</p>	<p>Previsto</p> <p>Exercício 14. Verifique se a função f definida no fragmento de VDM-SL que se segue,</p> <pre> types BTree = [Node]; Node :: item: int left: BTree right: BTree; functions f : int * BTree -> bool f(i,t) == cases t: nil -> false, mk_Node(x,l,r) -> if x = i then true else if (i < x) then f(i,l) else f(i,r) end; </pre> <p>está em condições de ser transformada num ciclo-while. Justifique adequadamente a sua resposta identificando eventuais lei de cálculo que tenha utilizado.</p> <p>□</p> <p style="text-align: right;">(v.s.f.f.)</p>

(cont.)

Previsto

Exercício 15. Recorde a lei (24) que representa estruturas indutivas sob a forma de pares (*heap*, *apontador*), bem como a respectiva função de abstracção (25). Seja $K \xleftarrow{f} K$ uma função de transformação de apontadores, usada como parâmetro na seguinte operação de re-alocação de células de um *heap* (vulg. *garbage collection*):

$$\begin{aligned} gcol & : (K \longrightarrow K) \longrightarrow (K \rightharpoonup F K) \longrightarrow (K \rightharpoonup F K) \\ gcol \ f \ M & \stackrel{\text{def}}{=} (F f) \cdot M \cdot f^\circ \end{aligned}$$

É de notar que $M \cdot f^\circ$ tem de ser simples, já que o contrário destruiria a simplicidade do *heap* resultante. Mas — será isso suficiente?

Uma *garbage collection* $gcol \ f \ M$ estará correcta se não destruir a representação, isto é, se

$$F(gcol \ f \ M, f \ k) = F(M, k) \quad (39)$$

se verificar, onde F é a abstracção (25). Complete o cálculo seguinte de uma condição que é suficiente para (39) estar garantida. E que condição é essa?

$$\begin{aligned} & F(gcol \ f \ M, f \ k) = F(M, k) \\ \equiv & \{ \dots \} \\ & \llbracket gcol \ f \ M \rrbracket (f \ k) = \llbracket M \rrbracket k \\ \equiv & \{ \dots \} \\ & \llbracket (F f) \cdot M \cdot f^\circ \rrbracket \cdot f = \llbracket M \rrbracket \\ \equiv & \{ \dots \} \\ & \llbracket in, (F f) \cdot M \cdot f^\circ \rrbracket \cdot f = \llbracket in, M \rrbracket \\ \Leftarrow & \{ \dots \} \\ & (F f) \cdot M \cdot f^\circ \cdot f = (F f) \cdot M \\ \Leftarrow & \{ \dots \} \\ & f^\circ \cdot f = id \end{aligned}$$

□

Exercício 16. Estude a semântica da construção

for id = e1 to e2 by e3 do s

(pág. 86 do manual de VDM-SL) e represente-a sob a forma de um hilomorfismo. □

(v.s.f.f.)

(cont.)

Previsto

Exercício 17. Complete a seguinte prova de (222), onde se assume S é simples:

$$\begin{aligned} & S \cdot R \subseteq T \\ \Rightarrow & \{ \dots \} \\ & (\ker S) \cdot R \subseteq S^\circ \cdot T \\ \Rightarrow & \{ \dots \} \\ & (\text{dom } S) \cdot R \subseteq S^\circ \cdot T \\ \Rightarrow & \{ \dots \} \\ & S \cdot (\text{dom } S) \cdot R \subseteq S \cdot S^\circ \cdot T \\ \Rightarrow & \{ \dots \} \\ & S \cdot R \subseteq T \end{aligned}$$

□

Exercício 18. Especifique em notação VDM-SL a conversa tarx de tar . Exprima-as como hilomorfismos, isto é, identifique S e S° no diagrama

$$\begin{array}{ccc} FS & \xrightarrow{\text{out}} & Id \rightarrow (File + FS) \\ \text{tar} \downarrow & & \downarrow \text{id} \rightarrow (\text{id} + \text{tar}) \\ Paths & \xleftarrow[S]{} & Id \rightarrow File + Paths \end{array}$$

□

Exercício 19. Aprecie, no modelo $ECTS$, a adição da relação topics (já anunciada na aula anterior) e o invariante topicsOk . Se tivesse que especificar que uma dada $u.\text{topics}$ é acíclica, como o faria? □

Exercício 20. Com base nas propriedades (237) e (233), prove que $[-, -]$ preserva abstrações e que $\langle -, - \rangle$ preserva representações. Isto é, mostre que $\langle R, S \rangle$ (resp. $[R, S]$) é uma relação de representação (resp. abstracção) sempre que R e S individualmente o são. □

O DOCENTE _____

Adenda aos sumários de MFP-I/0506

Contents

A	Lecture notes	101
A.1	On the “less defined than” ordering on models	101
B	Example of data refinement calculation	103
C	Relators	104
D	Parametric Polymorphism and “Theorems for Free”	105
E	Elements of the Fixpoint Calculus	107
E.1	Basic definitions	107
E.2	Computing fixpoints	108
E.3	Laws of the Fixpoint Calculus	108
E.3.1	Applications of μ -fusion theorem	109
F	Catalogue of Relational data refinement rules	200
F.1	Isomorphisms	200
F.2	Less than rules	200
F.3	Structural data refinement	201
G	PF-transform Reference Manual	202
G.1	Relational taxonomy	202
G.2	PF-transformation rules	202
G.3	Table of useful Galois connections	203
G.4	Other Galois connections	203
G.5	“Almost” Galois connections	203
G.6	Relational division	204
G.7	Meets	204
G.8	Splits	204
G.9	Eithers	204
G.10	Relational projection	205
G.11	Hylomorphisms	205
G.12	Iterative factorization of linear recursion	206
G.13	Catas and Anas	206
H	Solutions of some of the exercises	206

A Lecture notes

A.1 On the “less defined than” ordering on models

Suppose a component s of some piece of hardware gets faulty and needs to be replaced. Should no exact match be found off the shelf, the maintenance team will have to look around for *compatible* alternatives. What does *compatibility* mean in this context?

Let r be a candidate replacement for s and let the behaviour of both s and r be described by state-transition diagrams indicating, for each state a , the set of states reachable from a . So both s and r can be regarded as set-valued functions such that, for instance, component s may step from state a to state b iff $b \in (s\ a)$. Should $s\ a$ be empty, machine s will be in a deadlock and will fail.

The intuition behind r being a safe replacement for s — written $s \vdash r$ — is that not only r should not fail where s does not,

$$\langle \forall a : (s\ a) \supset \emptyset : \emptyset \subset (r\ a) \rangle$$

but also that it should behave “as s does”. Wherever $(s\ a)$ is nonempty, there is some freedom for r to behave within such a set of choices: r is allowed to be more deterministic than s . Altogether, one writes

$$s \vdash r \stackrel{\text{def}}{=} \langle \forall a : (s\ a) \supset \emptyset : \emptyset \subset (r\ a) \subseteq (s\ a) \rangle$$

cf (4) above.

Our first task will be to PF-transform (4). We first concentrate on transforming the test for non-deadlock states, which occurs twice in the formula, $(s\ a) \supset \emptyset$ and $(r\ a) \supset \emptyset$. A set is nonempty iff it contains at least one element. Therefore,

$$\begin{aligned} (s\ a) \supset \emptyset &\equiv \langle \exists x : x \in (s\ a) \rangle \\ &\equiv \{ \text{idempotence of } \wedge \} \\ &\quad \langle \exists x : x \in (s\ a) \wedge x \in (s\ a) \rangle \\ &\equiv \{ (214) \text{ twice and converse} \} \\ &\quad \langle \exists x : a(\in \cdot s)^\circ x \wedge x(\in \cdot s)a \rangle \\ &\equiv \{ \text{introduce } b = a ; \text{composition} \} \\ &\quad b = a \wedge b((\in \cdot s)^\circ \cdot (\in \cdot s))a \\ &\equiv \{ \text{introduce kernel} \} \\ &\quad b = a \wedge b(\mathbf{ker}(\in \cdot s))a \end{aligned}$$

Then we address the whole formula:

$$\begin{aligned} s \vdash r &\equiv \{ (4) \} \\ &\quad \langle \forall a : (s\ a) \supset \emptyset : \emptyset \subset (r\ a) \subseteq (s\ a) \rangle \\ &\equiv \{ \text{expand } \emptyset \subset (r\ a) \subseteq (s\ a) \} \\ &\quad \langle \forall a : (s\ a) \supset \emptyset : \emptyset \subset (r\ a) \wedge (r\ a) \subseteq (s\ a) \rangle \\ &\equiv \{ \text{expand tests for non-deadlock state and replace } (r\ a) \text{ by } (r\ b), \text{ cf. } b = a \} \\ &\quad \langle \forall a, b : b = a \wedge b(\mathbf{ker}(\in \cdot s))a : b = a \wedge b(\mathbf{ker}(\in \cdot r))a \wedge (r\ b) \subseteq (s\ a) \rangle \\ &\equiv \{ \text{dom } R = \mathbf{ker} R \cap id, \text{ for every } R [5, 3] \} \\ &\quad \langle \forall a, b : b(\text{dom}(\in \cdot s))a : b(\text{dom}(\in \cdot r))a \wedge (r\ b) \subseteq (s\ a) \rangle \\ &\equiv \{ \text{expand set-theoretic inclusion} \} \\ &\quad \langle \forall a, b : b(\text{dom}(\in \cdot s))a : b(\text{dom}(\in \cdot r))a \wedge \langle \forall c : c \in (r\ b) : c \in (s\ b) \rangle \rangle \\ &\equiv \{ (214) \text{ twice ; then introduce left-division (215)} \} \\ &\quad \langle \forall a, b : b(\text{dom}(\in \cdot s))a : b(\text{dom}(\in \cdot r))a \wedge b((\in \cdot r) \setminus (\in \cdot s))a \rangle \end{aligned}$$

$$\begin{aligned}
&\equiv \quad \{ \text{remove points ; relational inclusion and meet} \} \\
&\quad \mathbf{dom} (\in \cdot s) \subseteq \mathbf{dom} (\in \cdot r) \cap ((\in \cdot r) \setminus (\in \cdot s)) \\
&\equiv \quad \{ \text{remove membership by defining } R = \in \cdot r \text{ and } S = \in \cdot S \} \\
&\quad \mathbf{dom} S \subseteq \mathbf{dom} R \cap (R \setminus S)
\end{aligned}$$

Function s (resp. r) can be identified with the *power-transpose* [5, 6] of binary relation S (resp. R). Since transposition is an isomorphism, we can safely lift our original ordering on set-valued state-transition functions to state-transition relations and establish the relational PF-transform of (4) as follows:

$$S \vdash R \quad \equiv \quad \mathbf{dom} S \subseteq (R \setminus S) \cap \mathbf{dom} R \quad (101)$$

which converts to

$$S \vdash R \quad \equiv \quad (\mathbf{dom} S \subseteq \mathbf{dom} R) \wedge (R \cdot \mathbf{dom} S \subseteq S) \quad (102)$$

once Galois connections of meet (218) and left-division (217) are taken into account.

B Example of data refinement calculation

The BAMS example:

$$\begin{aligned}
 & AccId \multimap (2^{AccHolder} \times Amount) \\
 \cong_1 & \quad \{ (205) \} \\
 & AccId \multimap ((AccHolder \multimap 1) \times Amount) \\
 \cong_2 & \quad \{ \text{product swapping} \} \\
 & AccId \multimap (Amount \times (AccHolder \multimap 1)) \\
 \leq_3 & \quad \{ (208) \} \\
 & (AccId \multimap Amount) \times ((AccId \times AccHolder) \multimap 1) \\
 \cong_4 & \quad \{ (205) \} \\
 & (AccId \multimap Amount) \times 2^{AccId \times AccHolder}
 \end{aligned}$$

Abstractions and representations:

$$\begin{aligned}
 \cong_1 &= \begin{cases} f = id \multimap (\mathbf{dom} \times id) \\ r = id \multimap (set2fm \times id) \end{cases} \\
 \cong_2 &= \begin{cases} f = id \multimap swap \\ r = id \multimap swap \end{cases} \\
 \leq_3 &= \begin{cases} f = \bowtie_n \\ r = unnjoin \end{cases} \\
 \cong_4 &= \begin{cases} f = id \times set2fm \\ r = id \times \mathbf{dom} \end{cases}
 \end{aligned}$$

C Relators

Relators [4] are relational extensions of *functors*: $F A$ describes a parametric type while $F R$ is a relation from $F A$ to $F B$ provided R is a relation from A to B :

$$\begin{array}{ccc} A & \cdots & F A \\ R \downarrow & & \downarrow F R \\ B & \cdots & F B \end{array} \quad (103)$$

Relators have to be monotone and commute with composition, converse and the identity:

$$F (R \cdot S) = (F R) \cdot (F S) \quad (104)$$

$$F (R^\circ) = (F R)^\circ \quad (105)$$

$$F id = id \quad (106)$$

Examples The most simple relators are the *identity* relator Id , which is such that $Id A = A$ and $Id R = R$, and the *constant* relator K (for a particular concrete data type K) which is such that $K A = K$ and $K R = id_K$.

List maps extend to relators in the obvious way,

$$l(R^*)l' \equiv \text{length } l = \text{length } l' \wedge \forall i \in \text{inds } l. (l i) R (l' i) \quad (107)$$

Relators can also be multi-parametric. Two well-known examples of binary relators are product and sum,

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2 \rangle \quad (108)$$

$$R + S = [i_1 \cdot R, i_2 \cdot S] \quad (109)$$

where π_1, π_2 denote the projection functions of a Cartesian product, i_1, i_2 denote the injection functions of a disjoint union, and the *split/either* relational combinators are defined by (228) and (235), respectively.

By putting these four kinds of relator (product, sum, identity and constant) together with fixpoint definition one is able to specify a large class of parametric structures — called *polynomial* — such as those implementable in Haskell. For instance, the *Maybe* datatype is an implementation of polynomial relator $F = Id + 1$ (ie. $F A = A + 1$), where 1 denotes the *singleton* datatype, written $()$ in Haskell.

D Parametric Polymorphism and “Theorems for Free”

Parametric polymorphism: why?

- Less code (**specific** solution = **generic** solution + **customization**)
- Intellectual reward
- Last but not least, quotation (from *Theorems for free!*, by Philip Wadler [8]):

From the type of a polymorphic function we can derive a theorem that is satisfies. (...) How useful are the theorems so generated? Only time and experience will tell (...)

- No doubt: free theorems are **very** useful!

Reynolds abstraction theorem. The original theorem (the *abstraction theorem*) is due to J. Reynolds [7], and only later advertised by P. Wadler [8] under the “*theorem for free*” heading. We follow the pointfree styled presentation of this theorem in [2], which is remarkably elegant: let f be a polymorphic function $f : t \rightarrow t$, whose type t can be written according to the following “grammar” of types:

$$\begin{aligned} t &::= t' \leftarrow t'' \\ t &::= F(t_1, \dots, t_n) \quad \text{for } n\text{-ary relator } F \\ t &::= v \quad \text{for } v \text{ a type variable (= polymorphism “dimension”)} \end{aligned}$$

Let V be the set of type variables involved in type t ; $\{R_v\}_{v \in V}$ be a V -indexed family of relations (f_v in case all such R_v are functions); and R_t be a relation defined inductively as follows:

$$R_{t:=F(t_1, \dots, t_n)} = F(R_{t_1}, \dots, R_{t_n}) \quad (110)$$

$$R_{t:=v} = R_v \quad (111)$$

$$R_{t:=t' \leftarrow t''} = R_{t'} \leftarrow R_{t''} \quad (112)$$

where $R_{t'} \leftarrow R_{t''}$ is defined by Reynolds “arrow combinator”

$$g(S \leftarrow R)f \equiv g \cdot R \subseteq S \cdot f \quad \text{cf. diagram} \quad (113)$$

The *free theorem* of type t reads as follows:

given any function $f : t \rightarrow t$ and V as above, $f R_t f$ holds for any relational instantiation of type variables in V .

Note that this theorem is a result about t and holds for *any* polymorphic function of type t *independently* of its actual definition¹.

Example Let the target function be $f = \text{invl} : a^* \leftarrow a^*$. Calculation of $R_{t=a^* \leftarrow a^*}$ is as follows:

$$\begin{aligned} &R_{a^* \leftarrow a^*} \\ \equiv &\{ \text{rule (112)} \} \\ &R_{a^*} \leftarrow R_{a^*} \\ \equiv &\{ \text{rule (110)} \} \\ &R_{a^*}^* \leftarrow R_{a^*}^* \end{aligned}$$

¹See [2] for comprehensive evidence on the the power of this theorem when combined with Galois connections, which stems basically from the interplay between equations (2) and (3).

Calculation of FT $invl (R_{a^* \leftarrow a^*}) invl$ follows. The FT itself will predict (R_a abbreviated to R):

$$\begin{aligned} & invl(R^* \leftarrow R^*) invl \\ \equiv & \quad \{ \text{definition (113)} \} \\ & invl \cdot R^* \subseteq R^* \cdot invl \end{aligned}$$

In case R is a function r , the FT theorem boils down to $invl$'s **natural** property:

$$invl \cdot r^* = r^* \cdot invl$$

Further calculation (back to R):

$$\begin{aligned} & invl \cdot R^* \subseteq R^* \cdot invl \\ \equiv & \quad \{ \text{shunting rule} \} \\ & R^* \subseteq invl^\circ \cdot R^* \cdot invl \\ \equiv & \quad \{ \text{going pointwise} \} \\ & l(R^*)r \Rightarrow (invl \ l)(R^*)(invl \ r) \\ \equiv & \quad \{ \text{pointwise definition of } R^* \} \\ & \forall i \in inds \ l.(l \ i)R(r \ i) \Rightarrow (invl \ l)(R^*)(invl \ r) \end{aligned}$$

Natural transformations A collection of $G X \xleftarrow{T_X} F X$, for every X , is said to be a *natural transformation* from G to F iff, for every R ,

$$T \cdot F R \subseteq G R \cdot T \tag{114}$$

holds, cf. (omitting subscripts):

$$\begin{array}{ccccc} A & & G A & \xleftarrow{T} & F A \\ \downarrow R & & \downarrow G R & & \downarrow F R \\ B & & G B & \xleftarrow{T} & F B \end{array} \quad \supseteq$$

wherever T is a function, say t , then (114) equivaless

$$t(G R \leftarrow F R)t \equiv t \cdot F R \subseteq G R \cdot t \tag{115}$$

E Elements of the Fixpoint Calculus

E.1 Basic definitions

Definition 1 (Poset) A poset (A, \leq_A) is a set A equipped with a partial ordering \leq_A , that is, a relation $\leq_A \subseteq A \times A$ which is reflexive, transitive and antisymmetric.

□

Definition 2 (Pre/post-fixpoints) Let $A \xleftarrow{f} A$ be a (endo)function on poset (A, \leq_A) . Then

- every $a \in A$ such that

$$a \leq_A f a \quad (116)$$

is said to be a post-fixpoint of f .

- every $a \in A$ such that

$$a \geq_A f a \quad (117)$$

is said to be a pre-fixpoint of f .

- every $a \in A$ which is both a pre-fixpoint and a post-fixpoint of f is said to be a fixpoint of f and is such that

$$a = f a \quad (118)$$

holds.

□

Examples:

- Given endofunction

$$\begin{array}{ccc} f : [0, 10] & \rightarrow & [0, 10] \\ x & \rightsquigarrow & 10 - x \end{array}$$

one very easily checks that 5 is a fixpoint of f , since $f 5 = 10 - 5 = 5$.

- Let $R \subseteq P \times P$ be a relation on nonempty P in

$$x = R \cup R \cdot x \quad (119)$$

Define

$$f x = R \cup R \cdot x \quad (120)$$

on poset $(\mathcal{P}(P \times P), \subseteq)$. Then

- $P \times P$ is an example of a pre-fixpoint of f ($P \times P$ is the largest relation in the poset).
- \emptyset and R are examples of post-fixpoints of f . In fact, $\emptyset \subseteq R$ and $R \subseteq R \cup R^2$.

Clearly, every fixpoint $a = f a$ can be regarded as a “solution” to equation

$$x = f x \quad (121)$$

But one can also regard this equation as a “recursive” definition of its fixpoints. For instance, recall equation

$$x = 1 + \frac{x}{2}$$

The fact that 2 is a fixpoint of this equation can be rephrased to: “ $x = 1 + \frac{x}{2}$ ” is a recursive definition of number 2.

However, the following equation

$$x = \frac{x^2 + 3}{4}$$

admits two solutions (fixpoints) 1 e 3. What are we “recursively defining” here? The 1 or the 3? Furthermore, equation

$$x = x$$

defines any object! By contrast, some equations don’t have any solution at all. Think *eg.* of

$$x = x + 1$$

in \mathbb{N} . So, in this case, our recursive equation defines... nothing!

E.2 Computing fixpoints

Definition 3 (Monotone functions) A function $B \xleftarrow{f} A$ from poset (A, \leq_A) to poset (B, \leq_B) is said to be monotone iff

$$f \cdot \leq_A \subseteq \leq_B \cdot f \quad (122)$$

that is,

$$\forall a, a' \in A : a \leq_A a' \Rightarrow (f a) \leq_B (f a')$$

holds.

□

Theorem 1 (Lattice Fixpoints) [Tarski 1955]

Let

- $A \xleftarrow{f} A$ be a monotone function on a complete lattice $\langle A; \leq \rangle$;
- P be the set of all fixpoints of f , i.e.

$$P = \{a \in A \mid a = f a\}$$

Then

- P is non-empty and $\langle P; \leq \rangle$ is a complete (sub)lattice.
- In particular, the least of all fixpoints $(\bigwedge P)$ and the greatest one $(\bigvee P)$ are as follows:

$$\bigwedge P = \bigwedge \{x \mid x \geq f x\} \quad (123)$$

$$\bigvee P = \bigvee \{x \mid x \leq f x\} \quad (124)$$

We define:

$$\mu f \stackrel{\text{def}}{=} \bigwedge P \quad (125)$$

$$\nu f \stackrel{\text{def}}{=} \bigvee P \quad (126)$$

□

In the sequel we shall be focussing on *least* fixpoints.

E.3 Laws of the Fixpoint Calculus

Computation rule:

$$\mu f = f \mu f \quad (127)$$

Example: hylo-cancellation law

$$\llbracket R, S \rrbracket = R \cdot F \llbracket R, S \rrbracket \cdot S$$

Rolling rule:

$$\mu(g \cdot f) = g(\mu(f \cdot g)) \quad (128)$$

Example: the *hylo rolling rule*: Let $f = g \cdot h$ where $h X = F X \cdot S$ and $g = (R \cdot)$. Then

$$\begin{aligned} \mu f &= \mu(g \cdot h) = g(\mu(h \cdot g)) \\ &= \{ \text{definitions of } g, h \} \\ &\quad R \cdot (\mu X. (F(R \cdot X) \cdot S)) \\ &= \{ \text{relators} \} \\ &\quad R \cdot (\mu X. F R \cdot F X \cdot S) \end{aligned}$$

that is,

$$\llbracket R, S \rrbracket = R \cdot \llbracket F R, S \rrbracket$$

Square rule:

$$\mu f = \mu(f^2) \quad (129)$$

Monotonicity:

$$\mu f \leq \mu g \iff f \dot{\leq} g \quad (130)$$

where $\dot{\leq}$ is defined by (216).

Induction rule:

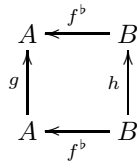
$$\mu f \leq x \iff f x \leq x \quad (131)$$

Diagonal rule: given monotonic $x \theta y$

$$\langle \mu x :: \langle \mu y :: x \theta y \rangle \rangle = \langle \mu x :: x \theta x \rangle \quad (132)$$

Last — but not least — μ -**fusion**:

Theorem 2 μ -fusion theorem *Let*



- h, g be monotonic,
- (A, \leq) and (B, \sqsubseteq) be complete **lattices**,
- f^b be a lower-adjoint.

Then

$$f^b(\mu h) = \mu g \iff f^b \cdot h = g \cdot f^b$$

E.3.1 Applications of μ -fusion theorem

Converse of a hylo

$$\llbracket S, R \rrbracket^\circ = \llbracket R^\circ, S^\circ \rrbracket \quad (133)$$

Proof: let $f^b = (_)\circ$ and

$$\begin{aligned} h X &= S \cdot F X \cdot R \\ g X &= T \cdot F X \cdot U \end{aligned}$$

that is,

$$\begin{aligned} \mu h &= \llbracket S, R \rrbracket \\ \mu g &= \llbracket T, U \rrbracket \end{aligned}$$

Then

$$\begin{aligned} &\llbracket S, R \rrbracket^\circ = \llbracket T, U \rrbracket \\ \Leftarrow &\quad \{ \mu\text{-fusion theorem} \} \\ &(S \cdot F X \cdot R)^\circ = T \cdot F (X^\circ) \cdot U \\ \equiv &\quad \{ \text{converse and } F \text{ is a relator} \} \\ &R^\circ \cdot F X^\circ \cdot S^\circ = T \cdot F X^\circ \cdot U \\ \Leftarrow &\quad \{ \text{Leibniz} \} \\ &R^\circ = T \wedge S^\circ = U \end{aligned}$$

Hylo(cata)-fusion:

$$V \cdot \llbracket S, R \rrbracket = \llbracket T, R \rrbracket \quad \Leftarrow \quad V \cdot S = T \cdot (F V) \quad (134)$$

Proof: since $(V \cdot) = (V \setminus)^b$,

$$\begin{aligned} & V \cdot \llbracket S, R \rrbracket = \llbracket T, R \rrbracket \\ \Leftarrow & \quad \{ \mu\text{-fusion theorem} \} \\ & V \cdot (S \cdot F X \cdot R) = T \cdot F (V \cdot X) \cdot R \\ \equiv & \quad \{ \text{associative } (\cdot) \text{ and relator } F \} \\ & (V \cdot S) \cdot F X \cdot R = T \cdot (F V) \cdot (F X) \cdot R \\ \Leftarrow & \quad \{ \text{Leibniz} \} \\ & V \cdot S = T \cdot (F V) \end{aligned}$$

Hylo(ana)-fusion:

$$\llbracket S, R \rrbracket \cdot V = \llbracket S, U \rrbracket \quad \Leftarrow \quad R \cdot V = F V \cdot U \quad (135)$$

Proof: $(\cdot V) = (/V)^b$. Then

$$\begin{aligned} & \llbracket S, R \rrbracket \cdot V = \llbracket S, U \rrbracket \\ \Leftarrow & \quad \{ \mu\text{-fusion theorem} \} \\ & (S \cdot F X \cdot R) \cdot V = S \cdot F (X \cdot V) \cdot U \\ \equiv & \quad \{ \text{associative } (\cdot) \text{ and relator } F \} \\ & S \cdot F X \cdot (R \cdot V) = S \cdot (F X) \cdot (F V) \cdot U \\ \Leftarrow & \quad \{ \text{Leibniz} \} \\ & R \cdot V = F V \cdot U \end{aligned}$$

F Catalogue of Relational data refinement rules

F.1 Isomorphisms

Product unit

$$A \times 1 \begin{array}{c} \xrightarrow{\pi_1} \\ \cong \\ \xleftarrow{\langle id, ! \rangle} \end{array} A \quad (201)$$

$$1 \times A \begin{array}{c} \xrightarrow{\pi_2} \\ \cong \\ \xleftarrow{\langle !, id \rangle} \end{array} A \quad (202)$$

“Maybe” transpose

$$(B + 1)^A \begin{array}{c} \xrightarrow{untot = (i_1^\circ \cdot)} \\ \cong \\ \xleftarrow{tot} \end{array} A \multimap B \quad (203)$$

that is,

$$f = tot \ R \equiv \langle \forall b, a :: (b \ R \ a \equiv (f \ a = i_1 \ b)) \rangle \quad (204)$$

Sets are multisets

$$2^A \begin{array}{c} \xrightarrow{set2fm} \\ \cong \\ \xleftarrow{dom} \end{array} A \multimap 1 \quad (205)$$

“Simple currying”

$$B \times C \multimap A \begin{array}{c} \xrightarrow{scurry} \\ \cong \\ \xleftarrow{\quad} \end{array} (C \multimap A)^B \quad (206)$$

where

$$f = \overline{R} \equiv \langle \forall a, b, c :: c \ (f \ a) \ b \equiv c \ R \ (a, b) \rangle$$

abbreviating *scurry* *R* to \overline{R} .

Relational either

$$(B + C) \multimap A \begin{array}{c} \xrightarrow{[-, \cdot]^\circ} \\ \cong \\ \xleftarrow{[-, \cdot]} \end{array} (B \multimap A) \times (C \multimap A) \quad (207)$$

F.2 Less than rules

“Nested join”

$$A \multimap (D \times (B \multimap C)) \begin{array}{c} \xrightarrow{unnjoin} \\ \leq \\ \xleftarrow{\bowtie_n} \end{array} (A \multimap D) \times ((A \times B) \multimap C) \quad (208)$$

where

$$\begin{aligned} R \bowtie_n S &= \langle R, \overline{S} \rangle \\ \text{unnjoin } R &= (\pi_1 \cdot R, \text{unpcurry}(\pi_2 \cdot R)) \end{aligned}$$

Concrete invariant induced by *unnjoin*:

$$\phi_{\text{unnjoin}}(M, N) = N \preceq M \cdot \pi_1$$

where

$$R \preceq S \equiv \text{dom } R \subseteq \text{dom } S \quad (209)$$

F.3 Structural data refinement

$$\begin{array}{c} A \begin{array}{c} \xrightarrow{R} \\ \leq \\ \xleftarrow{F} \end{array} B \quad \Rightarrow \quad F A \begin{array}{c} \xrightarrow{F R} \\ \leq \\ \xleftarrow{F F} \end{array} F B \end{array} \quad (210)$$

where F is an arbitrary relator (functor).

G PF-transform Reference Manual

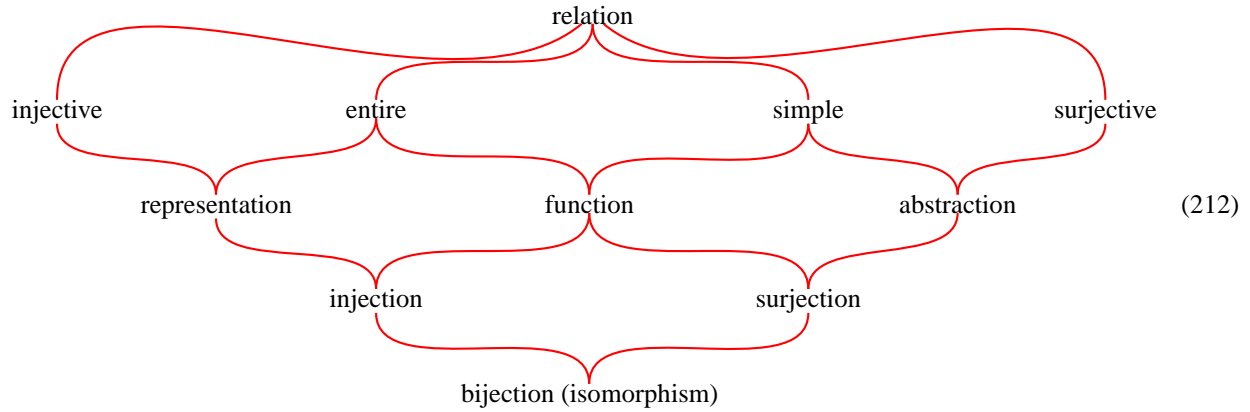
G.1 Relational taxonomy

Classification criteria:

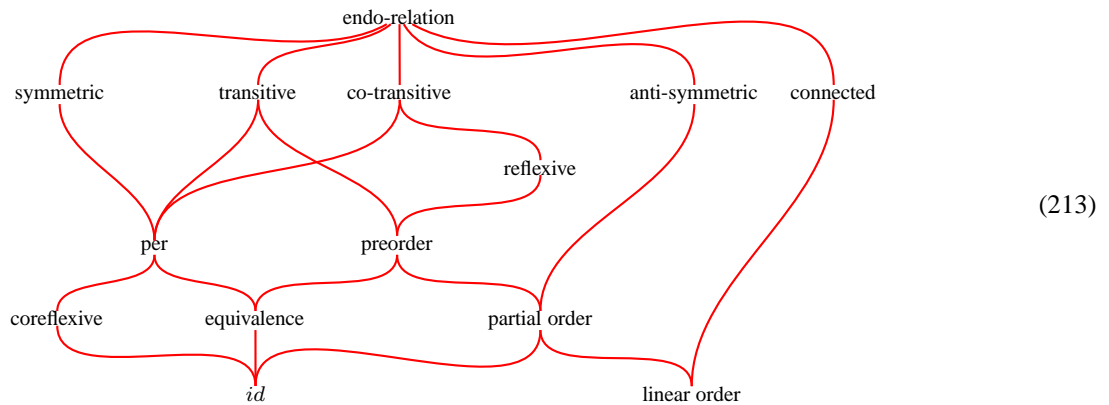
	<i>Reflexive</i>	<i>Coreflexive</i>
$\ker R$	entire R	injective R
$\text{img } R$	surjective R	simple R

(211)

Binary relations:



Orders:



G.2 PF-transformation rules

“Guardanapo”:

$$b(f^\circ \cdot R \cdot g)a \equiv (f b)R(g a) \quad (214)$$

Left-division:

$$b(R \setminus Y)a \equiv \langle \forall c : c R b : c Y a \rangle \quad (215)$$

Pointwise ordering on functions:

$$f \dot{\sqsubseteq} g \equiv f \sqsubseteq \sqsubseteq \cdot g \equiv \langle \forall a : (f a) \sqsubseteq (g a) \rangle \quad (216)$$

G.3 Table of useful Galois connections

Relational Operators as Galois Connections			
$(f \ X) \subseteq Y \equiv X \subseteq (g \ Y)$			
Description	$f = g^\flat$	$g = f^\sharp$	Obs.
converse	$(-)^{\circ}$	$(-)^{\circ}$	
shunting rule	$(f \cdot)$	$(f^{\circ} \cdot)$	NB: f is a function
“converse” shunting rule	$(\cdot f^{\circ})$	$(\cdot f)$	NB: f is a function
left-division	$(R \cdot)$	$(R \setminus \)$	R under ...
right-division	$(\cdot R)$	$(\ / R)$... over R
range	rng	$(\cdot \top)$	lower \subseteq restricted to coreflexives
domain	dom	$(\top \cdot)$	lower \subseteq restricted to coreflexives
implication	$(R \cap \)$	$(R \Rightarrow \)$	Note that $(R \Rightarrow) = (\neg R \cup \)$
difference	$(_ - R)$	$(R \cup \)$	
PROPERTIES			
cancellation	$X \subseteq (g \cdot f)X \qquad (f \cdot g)Y \subseteq Y$		
definition	$f \ X = \bigcap \{Y \mid X \subseteq gY\}$	$g \ Y = \bigcup \{X \mid f \ X \subseteq Y\}$	
distribution	$f(X \cup Y) = (f \ X) \cup (f \ Y)$	$g(X \cap Y) = (g \ X) \cap (g \ Y)$	$f(\bigcup_i X_i) = \bigcup_i (f \ X_i)$ $g(\bigcap_i X_i) = \bigcap_i (g \ X_i)$

(217)

G.4 Other Galois connections

Meet-universal

$$X \subseteq (R \cap S) \equiv (X \subseteq R) \wedge (X \subseteq S) \quad (218)$$

Join-universal

$$(R \cup S) \subseteq X \equiv (R \subseteq X) \wedge (S \subseteq X) \quad (219)$$

Split-universal

$$X \subseteq \langle R, S \rangle \equiv \pi_1 \cdot X \subseteq R \wedge \pi_2 \cdot X \subseteq S \quad (220)$$

Either-universal

$$X = [R, S] \equiv X \cdot i_1 = R \wedge X \cdot i_2 = S \quad (221)$$

G.5 “Almost” Galois connections

“Shunting” rules for S a simple relation:

$$S \cdot R \subseteq T \equiv (\text{dom } S) \cdot R \subseteq S^{\circ} \cdot T \quad (222)$$

$$R \cdot S^{\circ} \subseteq T \equiv R \cdot \text{dom } S \subseteq T \cdot S \quad (223)$$

G.6 Relational division

$$(R \setminus S) \cdot f = R \setminus (S \cdot f) \quad (224)$$

G.7 Meets

$$(S \cap T) \cdot R = (S \cdot R) \cap (T \cdot R) \iff T \cdot \text{img } R \subseteq T \vee S \cdot \text{img } R \subseteq S \quad (225)$$

Therefore, for f a function,

$$(S \cap T) \cdot f = (S \cdot f) \cap (T \cdot f) \quad (226)$$

$$R \cdot (S \cap T) = (R \cdot S) \cap (R \cdot T) \iff (\ker R) \cdot T \subseteq T \vee (\ker R) \cdot S \subseteq S \quad (227)$$

G.8 Splits

Definition equivalent to (220)

$$\langle R, S \rangle = \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \quad (228)$$

The same definition pointwise: for all a, b, c

$$(a, b) \langle R, S \rangle c \equiv a R c \wedge b S c \quad (229)$$

Split cancellation

$$\pi_1 \cdot \langle R, S \rangle = R \cdot \text{dom } S \quad \wedge \quad \pi_2 \cdot \langle R, S \rangle = S \cdot \text{dom } R \quad (230)$$

Split (conditional) fusion ²:

$$\langle R, S \rangle \cdot T = \langle R \cdot T, S \cdot T \rangle \iff R \cdot (\text{img } T) \subseteq R \vee S \cdot (\text{img } T) \subseteq S \quad (231)$$

Split absorption

$$\langle R \cdot T, S \cdot U \rangle = (R \times S) \cdot \langle T, U \rangle \quad (232)$$

Splits and converses:

$$\langle R, S \rangle^\circ \cdot \langle X, Y \rangle = (R^\circ \cdot X) \cap (S^\circ \cdot Y) \quad (233)$$

Therefore:

$$\ker \langle R, S \rangle = \ker R \cap \ker S \quad (234)$$

G.9 Eithers

Definition:

$$[R, S] = (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \quad (235)$$

From (221), all coproduct properties extend to relations, in particular: $+$ -reflexion:

$$id = [i_1, i_2] \quad (236)$$

etc. Eithers and converses:

$$[R, S] \cdot [T, U]^\circ = (R \cdot T^\circ) \cup (S \cdot U^\circ) \quad (237)$$

²Theorem 12.30 in [1].

G.10 Relational projection

Definition

$$\pi_{g,f}R \stackrel{\text{def}}{=} g \cdot R \cdot f^\circ \quad (238)$$

Property

$$\pi_{g,f}R \subseteq S \equiv g(S \leftarrow R)f \quad (239)$$

G.11 Hylomorphisms

Definition

$$\llbracket R, S \rrbracket \stackrel{\text{def}}{=} \langle \mu X :: R \cdot (F X) \cdot S \rangle \quad (240)$$

Cancellation

$$\llbracket R, S \rrbracket = R \cdot F \llbracket R, S \rrbracket \cdot S \quad (241)$$

Hylomorphism converse

$$\llbracket R, S \rrbracket^\circ = \llbracket S^\circ, R^\circ \rrbracket \quad (242)$$

Hylo(cata)-fusion

$$V \cdot \llbracket S, R \rrbracket = \llbracket T, R \rrbracket \Leftarrow V \cdot S = T \cdot (F V) \quad (243)$$

Hylo(ana)-fusion

$$\llbracket S, R \rrbracket \cdot V = \llbracket S, U \rrbracket \Leftarrow R \cdot V = F V \cdot U \quad (244)$$

Monotonicity:

$$\llbracket T, U \rrbracket \subseteq \llbracket R, S \rrbracket \Leftarrow T \subseteq R \wedge U \subseteq S \quad (245)$$

Induction

$$\llbracket R, S \rrbracket \subseteq T \Leftarrow R \cdot F T \cdot S \subseteq T \quad (246)$$

Coreflexion:

$$\llbracket R, S \rrbracket \subseteq id \Leftarrow R \cdot S \subseteq id \quad (247)$$

Corollaries:

$$(in \cdot \Phi) \subseteq id \Leftarrow \Phi \subseteq id \quad (248)$$

$$\llbracket R, R^\circ \rrbracket \subseteq id \Leftarrow R \text{ is simple} \quad (249)$$

The while loop hylomorphism:

$$\underline{\text{while}}\ p\ \underline{\text{do}}\ l = \llbracket [id, id], (id + l) \cdot (\neg \cdot p)? \rrbracket$$

that is

$$\underline{\text{while}}\ p\ \underline{\text{do}}\ l = \langle \mu f :: (\neg \cdot p) \rightarrow id, f \cdot l \rangle \quad (250)$$

Linear recursion (functional) hylomorphism:

$$\begin{aligned} f &: A \longrightarrow C \\ f &= p \rightarrow b, \theta \cdot \langle d, f \cdot e \rangle \end{aligned} \quad (251)$$

G.12 Iterative factorization of linear recursion

First law for while loop conversion: for θ associative, the following equality holds:

$$\begin{aligned} \langle \mu f : p \rightarrow b, \theta \cdot \langle d, f \cdot e \rangle \rangle &= p \rightarrow b, \theta \cdot (id \times b) \cdot w \cdot \langle d, e \rangle \\ &\text{where} \\ w &= \underline{while} (\neg \cdot p \cdot \pi_2) \underline{do} \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle \end{aligned} \quad (252)$$

Second law for while loop conversion: for (θ, u) a monoid, the following equality holds:

$$\langle \mu f : p \rightarrow \underline{u}, \theta \cdot \langle d, f \cdot e \rangle \rangle = \pi_1 \cdot w \cdot \langle \underline{u}, id \rangle \quad (253)$$

where w is the same as in (252) above.

G.13 Catas and Anas

Definitions

$$\langle R \rangle = \llbracket R, in^\circ \rrbracket \quad (254)$$

$$\langle S \rangle = \llbracket in, S \rrbracket \quad (255)$$

Converse of *ana* is *cata* of converse

$$\langle S \rangle^\circ = \langle S^\circ \rangle \quad (256)$$

Mutual-recursion law (also called “Fokkinga law”):

$$\begin{cases} f \cdot in = h \cdot F \langle f, g \rangle \\ g \cdot in = k \cdot F \langle f, g \rangle \end{cases} \equiv \langle f, g \rangle = \langle \langle h, k \rangle \rangle \quad (257)$$

Corollary (“banana-split”):

$$\langle \langle i \rangle, \langle j \rangle \rangle = \langle \langle i \cdot F \pi_1, j \cdot F \pi_2 \rangle \rangle$$

that is

$$\langle \langle i \rangle, \langle j \rangle \rangle = \langle (i \times j) \cdot \langle F \pi_1, F \pi_2 \rangle \rangle \quad (258)$$

H Solutions of some of the exercises

Resolução 1: Proof that f (resp. r) in (1) is surjective (resp. injective): from the equation infer $r \subseteq f^\circ$ which,

- together with $f \subseteq f$ we get $f \cdot r \subseteq f \cdot f^\circ$, that is, $id \subseteq f \cdot f^\circ$, which means that the image of f is reflexive and so f is surjective.
- together with $r^\circ \subseteq r^\circ$ leads to

$$\begin{aligned} r^\circ \cdot r &\subseteq r^\circ \cdot f^\circ \\ &\equiv \{ \} \\ r^\circ \cdot r &\subseteq (f \cdot r)^\circ \\ &\equiv \{ \} \\ r^\circ \cdot r &\subseteq id \end{aligned}$$

and so r is injective.

□

Resolução 7: O teorema é o mesmo para *sort* e *nub*, uma vez que têm o mesmo tipo. Quer dizer, onde nos cálculos que se seguem ocorre *sort*, pode substituir-se por *nub*: Ter-se-á

$$\begin{aligned}
& \text{sort}(R_{(a^* \leftarrow a^*) \leftarrow (2 \leftarrow (a \times a))}) \text{sort} \\
\equiv & \quad \{ (110, 111, 112) ; R_{t:=2} = \text{id} \text{ (cf. functor constante)} \} \\
& \text{sort}((R^* \leftarrow R^*) \leftarrow (\text{id} \leftarrow (R \times R))) \text{sort} \\
\equiv & \quad \{ (113) \} \\
& \text{sort} \cdot (\text{id} \leftarrow (R \times R)) \subseteq (R^* \leftarrow R^*) \cdot \text{sort} \\
\equiv & \quad \{ \text{shunting} \} \\
& (\text{id} \leftarrow (R \times R)) \subseteq \text{sort}^\circ \cdot (R^* \leftarrow R^*) \cdot \text{sort} \\
\equiv & \quad \{ \text{introdução das variáveis } f \text{ e } g \} \\
& f(\text{id} \leftarrow (R \times R))g \Rightarrow (\text{sort } f)(R^* \leftarrow R^*)(\text{sort } g) \\
\equiv & \quad \{ (110, 112, 113) \} \\
& f \cdot (R \times R) \subseteq g \Rightarrow (\text{sort } f) \cdot R^* \subseteq R^* \cdot (\text{sort } g)
\end{aligned}$$

Caso em que R é uma função r :

$$\begin{aligned}
& f \cdot (r \times r) = g \Rightarrow (\text{sort } f) \cdot r^* = r^* \cdot (\text{sort } g) \\
\equiv & \quad \{ \text{introduzindo variáveis} \} \\
& \langle \forall a, b : : f(r a, r b) = g(a, b) \rangle \Rightarrow \langle \forall l : : (\text{sort } f)(r^* l) = r^*(\text{sort } g l) \rangle
\end{aligned}$$

Exprimindo os predicados binários f, g pelas ordens \leq, \preceq (notação infixa), ter-se-á:

$$\langle \forall a, b : : r a \leq r b \equiv a \preceq b \rangle \Rightarrow \langle \forall l : : (\text{sort } (\leq))(r^* l) = r^*(\text{sort } (\preceq) l) \rangle$$

Quer dizer, se r for monótona e injectiva, então

$$\text{sort } (\leq) [r a \mid a \leftarrow l]$$

é a mesma lista que

$$[r a \mid a \leftarrow \text{sort } (\preceq) l]$$

□

References

- [1] Chritiene Aarts, Roland Backhouse, Paul Hoogendijk, Ed Voermans, and Jaap van der Woude. A relational theory of datatypes, December 1992. Available from www.cs.nott.ac.uk/~rcb/papers.
- [2] K. Backhouse and R.C. Backhouse. Safety of abstract interpretations for free, via logical relations and Galois connections. *Science of Computer Programming*, 2003. Accepted for publication.
- [3] R.C. Backhouse. *Mathematics of Program Construction*. Univ. of Nottingham, 2004. Draft of book in preparation. 608 pages.
- [4] R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude. Polynomial relators. In *2nd Int. Conf. Algebraic Methodology and Software Technology (AMAST'91)*, pages 303–362. Springer LNCS, 1992.
- [5] R. Bird and O. de Moor. *Algebra of Programming*. Series in Computer Science. Prentice-Hall International, 1997. C.A. R. Hoare, series editor.
- [6] J.N. Oliveira and C.J. Rodrigues. Transposing relations: from *Maybe* functions to hash tables. In *MPC'04 : Seventh International Conference on Mathematics of Program Construction, 12-14 July, 2004, Stirling, Scotland, UK (Organized in conjunction with AMAST'04)*, volume 3125 of *Lecture Notes in Computer Science*, pages 334–356. Springer, 2004.

- [7] J.C. Reynolds. Types, abstraction and parametric polymorphism. *Information Processing* 83, pages 513–523, 1983.
- [8] P. Wadler. Theorems for free! In *4th International Symposium on Functional Programming Languages and Computer Architecture*, London, Sep. 1989. ACM.
- [9] N. Wirth. *Algorithms + Data Structures = Programs*. Prentice-Hall, 1976.