Universidade do Minho

2006/2007	1.º Semestre ☐ 2.º Semestre ☒ Anual ☐
DISCIPLINAS Métodos Formais de Programação II (7008N2) + Opção II — Métodos Formais de Programação II (5308P3) CURSOS LMCC + LESI	DOCENTE J.N. Oliveira – 406006

AULA	SUMÁRIO
Teórica 2007.03.01 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Não houve aula (ausência do docente em reunião de serviço na Universidade de Aveiro). O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.03.05 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	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`. Breve introdução ao VDM++, que deverá ser usado na parte prática da disciplina. Noção de *objectificação* de uma especificação puramente funcional. Exemplo: os modelos `stackAlg.vpp` e `stackObj.vpp`. O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.03.08 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	*Introdução:* De volta ao ciclo de vida de Balzer e ao binómio *especificação / implementação*. Relações como especificações e funções como implementações. Necessidade de *realizar* (reificar) especificações. Uso alternativo do termo *refinar* em lugar de reificar. Princípio da reificação de um par **pre**/**post** por uma função f [a]: $$\langle \forall\ a\ ::\ pre\ a \Rightarrow post(f\ a, a)\rangle \quad (1)$$ isto é, $$f \cdot Pre \subseteq Post \quad (2)$$ após transformação-PF, para $Pre = \lceil pre \rceil$ e $Post = \lceil post \rceil$. Generalização do princípio anterior à satisfação de uma qualquer relação binária S por uma função f do mesmo tipo: $$S \vdash f \stackrel{\text{def}}{=} f \cdot \delta\, S \subseteq S \quad (3)$$ *(v.s.f.f.)* [a] Trata-se da obrigação de prova 3.1 do livro *Systematic Software Development Using VDM* [4], página 51.

(cont.)	Cálculo de (2) a partir de (3), para $S = Post \cdot Pre$. Interpretação destes conceitos no contexto do VDM++. A omnipresença das variáveis de instância. Noção de *estado interno* de um modelo VDM++. Semântica relacional de uma operação (**pre**/**post**) em VDM++. O DOCENTE _____

AULA	SUMÁRIO

Prática
2007.03.12
2.ª-feira, 16h00–18h00
DI-A.2 (LMCC+LESI)
	Resolução dos exercícios seguintes:
	
	Exercício 1. Uma especificação S diz-se "total" sempre que S é inteira. Mostre que, para essas especificações, se tem
	
	$$S \vdash f \quad \equiv \quad f \subseteq S \tag{4}$$
	
	□
	
	Exercício 2. Uma especificação S diz-se "funcional" sempre que S é simples. Mostre que, para essas especificações, se tem
	
	$$S \vdash f \quad \equiv \quad f \cdot S^{\circ} \subseteq id \tag{5}$$
	
	□
	
	Exercício 3. Provar que a especificação que se segue, escrita em VDM-SL, $S \vdash id$
	
	`S(n: real) r: real`
	`pre n > 1`
	`post r*r + 2*n*n = 3*n*r;`
	
	é satisfeita pela função identidade. Será id a única implementação funcional de S? Justificar informalmente. □
	
	Exercício 4. Mostrar, recorrendo à transformada-PF, que a função
	
	$$abs\, i \quad \stackrel{\text{def}}{=} \quad if\, i < 0\, then\, -i\, else\, i \tag{6}$$
	
	satisfaz a especificação
	
	$$S(i : \mathbb{Z})\ r : \mathbb{Z}$$
	$$\textbf{post}\ \ r = i \vee r = -i$$
	
	Sugestão: mostrar que a transformada-PF de S é $id \cup sym$, onde $sym\, i \stackrel{\text{def}}{=} -i$, e recorrer ao condicional de McCarthy
	
	$$R \rightarrow S\ ,\ T \quad \stackrel{\text{def}}{=} \quad (S \cdot \delta\, R) \cup (T \cdot \neg \delta\, R) \tag{7}$$
	
	para transformar abs. □
	
	(v.s.f.f.)

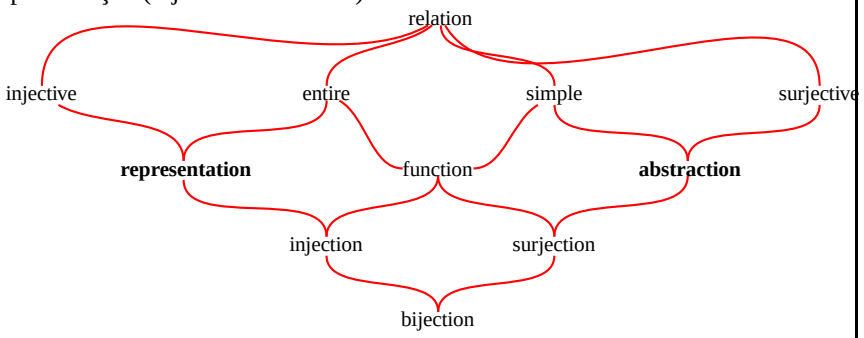
(cont.)	**Exercício 5.** Demonstrar, a partir (3), as seguintes propriedades da relação $\vdash$: $$\begin{aligned} \bot \vdash f \quad &, \quad \top \vdash f && (8)\\ (\ker g) \vdash f \quad &\equiv \quad g \cdot f = g && (9)\\ g \vdash f \quad &\equiv \quad f = g && (10)\\ (S \cup R) \vdash f \quad &\Leftarrow \quad S \vdash f \ \wedge \ R \vdash f && (11)\\ (S \cap R) \vdash f \quad &\Leftarrow \quad S \vdash f \ \wedge \ R \vdash f && (12)\\ S \vdash f \cup R \quad &\Leftarrow \quad S \vdash f \ \wedge \ \delta\, R \subseteq \delta\, S && (13) \end{aligned}$$ $\square$ O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.03.15 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Dois caminhos para generalizar (3) ainda mais: (a) reificação entre especificações por aumento simultâneo de *definição* e de *determinismo*, $$S \vdash R \quad \equiv \quad (\delta\, S \subseteq \delta\, R) \ \wedge \ (R \cdot \delta\, S \subseteq S) \qquad (14)$$ sendo fácil ver que, para $R := f$, (14) se reduz a (3); (b) reificação envolvendo *mudança de tipos de dados*, cf. $$S \vdash f \cdot R \cdot r \qquad (15)$$ para o diagrama $$\begin{array}{ccc} B & \xleftarrow[S]{} & A \\ {\scriptstyle f}\uparrow & & \downarrow{\scriptstyle r} \\ B_1 & \xleftarrow[R]{} & A_1 \end{array}$$ Interpretação das funções f e r. Noção de *abstracção* e de *representação* de dados. Refinamento de dados versus *conversão de formatos*. Aspectos práticos: *migração* e *canalização* de dados sem perda de informação. Caso mais simples: formatos isomorfos. Relaxe da relação de isomorfismo $A \cong A_1$ à relação $A \leq A_1$ captando o facto de A ter menos informação que A_1. Exemplo: representação de conjuntos de naturais por sequências ordenadas sem elementos repetidos. Assim: isomorfismo f tal que $f \cdot f^\circ = id$ e $f^\circ \cdot f = id$ generalizado a f e r tal que $f \cdot r = id$, deixando de ser exigido $r \cdot f = id$. Demonstração do facto $$f \cdot r \quad = \quad id \qquad \Rightarrow \qquad f \text{ é sobrejectiva e } r \text{ é injectiva} \qquad (16)$$ a partir de $r \subseteq f^\circ$ e do seu converso: *(v.s.f.f.)*

(cont.)

$$
\begin{array}{ll}
 & f \cdot r \;=\; id \\
\equiv & \quad \{ \text{ igualdade de funções ; shunting ; conversos } \} \\
 & r \subseteq f^\circ \;\wedge\; r^\circ \subseteq f \\
\Rightarrow & \quad \{ \text{ monotonia da composição } \} \\
 & f \cdot r \subseteq f \cdot f^\circ \;\wedge\; r^\circ \cdot r \subseteq f \cdot r \\
\equiv & \quad \{ \text{ igualdade de que se partiu } \} \\
 & id \subseteq f \cdot f^\circ \;\wedge\; r^\circ \cdot r \subseteq id \\
\equiv & \quad \{ \text{ definições } \} \\
 & f \text{ é sobrejectiva } \wedge\; r \text{ é injectiva}
\end{array}
$$

Generalização a relações de abstracção (simples + sobrejectivas) e de representação (injectivas e inteiras) — de volta à taxonomia



O requisito de invertibilidade

$$F \cdot R = id \tag{17}$$

e sua interpretação por "ping-pong": da direita para a esquerda ($id \subseteq F \cdot R$) significa que todo o valor abstracto é representável; da esquerda para a direita ($F \cdot R \subseteq id$) significa que é sempre possível recuperar de um valor concreto o valor abstracto que ele representa.

O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.03.19 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Início do primeiro caso de estudo de reificação em VDM: a implementação da álgebra de conjuntos finitos sobre tabelas de *hashing*. Conjectura da implementação da operação de *procura* de um elemento e das funções de abstracção e de representação envolvidas. *(v.s.f.f.)*

(cont.)	Resolução dos exercícios seguintes: **Exercício 6.** Mostre que a equivalência (5) se reduz a uma implicação $$S \vdash f \quad \Leftarrow \quad f \cdot S^\circ \subseteq id \tag{18}$$ no caso de S ser qualquer. **Sugestão**: recorra a (162). □ **Exercício 7.** Provar que a especificação: $$Abs(i : \mathbb{Z})\ r : \mathbb{Z}$$ $$\textbf{post}\ \ 0 \leq r\ \wedge\ (r = i \vee r = -i)$$ é satisfeita por abs (6) [a]. **Sugestão**: prosseguir com a transformada-PF do Ex. 4 e recorra a (18). □ **Exercício 8.** Mostrar que, ao fim e ao cabo, $abs = Abs$. □ O DOCENTE _____ [a] Este é o Ex.3.2.4 de [4], pág. 59.

AULA	SUMÁRIO
Teórica 2007.03.22 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Introdução ao estudo de um *cálculo de reificação de dados*. Manutenção do requisito de invertibilidade (17) por *inequações de representação* $$A \underset{F}{\overset{R}{\leq}} B \tag{19}$$ sempre que o par de testemunhas R e F é tal que R é representação, F é abstracção e $R \subseteq F^\circ$ (requisito de conexão entre elas). Demonstração de que, sempre que (19) se verifica, a invertibilidade (17) está garantida: $$\begin{array}{ll} & F \cdot R = id \\ \equiv & \quad \{\ \text{igualdade de relações}\ \} \\ & F \cdot R \subseteq id\ \wedge\ id \subseteq F \cdot R \\ \equiv & \quad \{\ \text{img}\ F = id\ \text{e}\ \ker R = id\ \} \\ & F \cdot R \subseteq F \cdot F^\circ\ \wedge\ R^\circ \cdot R \subseteq F \cdot R \\ \equiv & \quad \{\ \text{conversos}\ \} \\ & F \cdot R \subseteq F \cdot F^\circ\ \wedge\ R^\circ \cdot R \subseteq R^\circ \cdot F^\circ \\ \Leftarrow & \quad \{\ \text{monotonia de}\ (F\cdot)\ \text{e}\ (R^\circ\cdot)\ \} \\ & R \subseteq F^\circ\ \wedge\ R \subseteq F^\circ \end{array}$$ *(v.s.f.f.)*

(cont.)	$\equiv$ { trivia } $R \subseteq F^\circ$ $\equiv$ { por hipótese } TRUE Primeiros exemplos (óbvios) de inequações de representação envolvendo produtos e coprodutos: $A \overset{R}{\underset{\pi_1}{\leq}} A \times B$ e $A \overset{i_1}{\underset{F}{\leq}} A + B$ (20) Cálculo simples de R e F para estes exemplos, baseado em leis de fusão. Noção de *invariante concreto* induzido por uma inequação de representação: trata-se do predicado determinado pela coreflexiva $\Phi = id \cap R \cdot F$ (21) cuja interpetação para o caso funcional é $\phi\, b \equiv r(f\ b) = b$ (cf. invertibilidade). O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.03.26 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Não houve aula (envolvimento do docente na organização da conferência ETAPS'07). O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.03.29 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Não houve aula (envolvimento do docente na organização da conferência ETAPS'07). O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.04.12 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Início do estudo de um catálogo de leis de refinamento de dados. O isomorfismo como caso particular de refinamento: revisão dos principais isomorfismos envolvendo produtos, coprodutos e funções (exponenciais) *(v.s.f.f.)*

(cont.)	Abordagem relacional ao refinamento de estruturas como listas, *mappings*, etc. Uso da notação $A \to B$ para designar o tipo de todas as relações binárias "de A para B", e da notação $A \rightharpoonup B$ para designar o tipo de todas as relações *simples* "de A para B", logo incluindo o tipo `map` A `to` B em VDM-SL. Apresentação dos isomorfismos de transposição de relações: $$A \to B \underset{(\in \cdot)}{\overset{\Lambda}{\cong}} (\mathcal{P}B)^A$$ isto é $$f = \Lambda R \quad \equiv \quad R = \in \cdot f \qquad (22)$$ (qualquer relação pode ser transformada numa função para conjuntos), $$(C \to A)^B \underset{(\tilde{-})}{\overset{(-)^\circ}{\cong}} B \times C \to A$$ — corolário de (22) — e, para relações simples, a transposição-$Maybe$: $$untot = (i_1^\circ \cdot\ )$$ $$(B+1)^A \underset{tot}{\overset{untot}{\cong}} A \rightharpoonup B$$ isto é $$f = tot\ M \quad \equiv \quad M = i_1^\circ \cdot f \qquad (23)$$ Início do estudo das leis envolvendo $\leq$. O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.04.16 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Resolução dos exercícios seguintes: **Exercício 9.** Demonstrar as propriedades seguintes das transposições (22) e (23): $$\in \cdot (\Lambda R) = R \qquad (24)$$ $$\Lambda(R \cdot S) = (\Lambda R) \cdot S \Leftarrow (\Lambda R) \cdot S \text{ é função} \qquad (25)$$ $$i_1^\circ \cdot (tot\ M) = M \qquad (26)$$ $$tot(M \cdot N) = (tot\ M) \cdot N \Leftarrow (\Lambda R) \cdot S \text{ é função} \qquad (27)$$ □ *(v.s.f.f.)*

(cont.)

Exercício 10. O diagrama que se segue documenta o processo de representação de um conjunto S (visto como uma coreflexiva) por uma tabela de *hashing* t,

$$t \quad = \quad \Lambda(S \cdot h^\circ) \qquad \begin{array}{ccc} A & \xleftarrow{S} & A \\ {\scriptstyle\in}\uparrow & & \uparrow{\scriptstyle h^\circ} \\ \mathcal{P}A & \xleftarrow[t]{} & B \end{array} \tag{28}$$

em que h é uma dada função de *hashing*. Complete o seguinte raciocínio que explica o significado desta representação:

$$\begin{array}{ll} & t \quad = \quad \Lambda(S \cdot h^\circ) \\ \equiv & \quad \{\ldots\} \\ & \in \cdot t \quad = \quad S \cdot h^\circ \\ \equiv & \quad \{\ldots\} \\ & a(S \cdot h^\circ)b \quad \equiv \quad a(\in \cdot t)b \\ \equiv & \quad \{\ldots\} \\ & aSa' \ \wedge \ a'h^\circ b \quad \equiv \quad a \in (t\ b) \\ \equiv & \quad \{\ldots\} \\ & a \in S \ \wedge \ a = a' \ \wedge \ b = h\ a' \quad \equiv \quad a \in (t\ b) \\ \equiv & \quad \{\ldots\} \\ & a \in S \ \wedge \ b = h\ a \quad \equiv \quad a \in (t\ b) \\ \equiv & \quad \{\ldots\} \\ & a \in S \quad \equiv \quad a \in t(h\ a) \end{array}$$

□

O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.04.19 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Estudo detalhado das leis de $\leq$-refinamento que constam do apêndice C. Cálculo da conecção de Galois asscociada à lei $A \to B \underset{(\in\cdot)}{\overset{collect}{\leq}} A \rightharpoonup \mathcal{P}B$ (29) O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.04.23 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Resolução do seguinte exercício, como preparação para o trabalho prático da disciplina: **Exercício 11.** Especificar sobre os modelos FS (40) e Tar (41) a função $mkdir$ que cria uma nova directoria na directoria corrente. (NB: atenção ao invariante sobre Tar.) □ O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.04.26 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	*Refinamento de dados por cálculo (continuação) :* Propriedades da relação $\leq$: reflexividade e transitividade e suas provas. Relacionadores (*relators*) e o refinamento estruturado. O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.04.30 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Análise detalhada do cálculo da implementação em SQL do modelo PPD da disciplina de *MFP-I.* Conclusão do estudo sobre *tabelas de hashing*: **Exercício 12.** Defina-se $repf\ S = collect(S \cdot h^\circ)$, onde S é a (coreflexiva que modela) um conjunto e h é uma função de *hashing*. Complete o seguinte processo de cálculo que mostra que $refp$ é a função que se definiu em VDM-SL em (39): $$\begin{array}{ll} & repf\ S \\ = & \quad \{\ldots\} \\ & \Lambda(S \cdot h^\circ) \cdot \delta\,(S \cdot h^\circ) \\ = & \quad \{\ldots\} \\ & \Lambda(S \cdot h^\circ) \cdot \mathsf{ker}\,(S \cdot h^\circ) \\ = & \quad \{\ldots\} \\ & \Lambda(S \cdot h^\circ \cdot h) \cdot S \cdot h^\circ \\ = & \quad \{\ (178)\ \} \\ & \pi_{\Lambda(S \cdot h^\circ \cdot h),h} S \\ = & \quad \{\ldots\} \\ & \vdots \end{array}$$ □ O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.05.03 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Introdução ao refinamento de modelos de dados recursivos. Teorema de desrecursivação genérica: $$\mu\mathsf{F} \quad \leq \quad (K \rightharpoonup \mathsf{F}\,K) \times K \qquad (30)$$ Exemplos: desrecursivação do tipo de dados $Exp \cong A + B \times Exp^{\star}$. Primeira análise da desrecursivação do modelo $$\begin{array}{rrll} GenDia :: & indiv : & token & \text{/*data about an individual */} \\ & mother : & [GenDia] & \\ & father : & [GenDia] & \end{array} \qquad (31)$$ Necessidade de ***hilomorfismos*** relacionais para exprimir abstracções e representações. O hilomorfismo $[\![\, R, S \,]\!]$ como solução da equação relacional $$X \quad = \quad R \cdot \mathsf{F}\,X \cdot S$$ Exemplos introdutórios: "fold" de conjuntos e de *mappings*. O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.05.07 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	1ª parte [a]: *Introdução ao cálculo de pontos-fixos.* Funções monótonas, pré/pós-pontos-fixos. Teorema de Tarski. Notação μ. Resolução de equações relacionais. Casos típicos: hilo-equações $X = \underbrace{R \cdot (\mathsf{F}\,X) \cdot S}_{f\ X}$ e outras, por exemplo, $$X = \underbrace{R \cup R \cdot X}_{g\ X}$$ (cf. fecho transitivo.) 2ª parte: foi feito o ponto da situação quanto aos trabalhos práticos. O DOCENTE _____ [a] Devido à realização das JOIN'07, a primeira parte desta aula foi teórica.

AULA	SUMÁRIO
Teórica 2007.05.10 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Não houve aula (tolerância das JOIN'07). O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.05.14 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Não houve aula (tolerância do Enterro da Gata). O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.05.17 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Não houve aula (tolerância do Enterro da Gata). O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.05.21 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Caso de estudo sobre a lei (30): formulação (em VDM-SL) da relação de abstracção envolvida na aplicação dessa lei ao tipo (31) (ver secções B.4 e G). Expressão dessa relação sobre a forma de um anamorfismo relacional. Cálculo da relação de accessibilidade e sua expressão em VDM-SL. Foi ainda feito o ponto da situação quanto aos trabalhos práticos. O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.05.24 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Estudo do cálculo de pontos fixos (secção F, pág. 33). Teorema da *fusão-*μ (119) e sua aplicação ao cálculo de hilomorfismos relacionais. Os tipos colectivos de dados em VDM-SL e seus hilomorfismos. O tipo `map` A `to` B. O tipo `set of` A e a notação $\{\!|g|\!\}$. Catamorfismos e anamorfismos relacionais. Estudo da abstracção associada a (30) $$y\ F(H,k) \quad \equiv \quad y[\![H]\!]k \qquad (32)$$ como anamorfismo relacional. O DOCENTE _____

AULA	SUMÁRIO
Prática 2007.05.28 2.ª-feira, 16h00–18h00 DI-A.2 (LMCC+LESI)	Resolução dos exercícios seguintes: **Exercício 13.** Considere o raciocínio $Books \quad = \quad ISBN \rightharpoonup Title \times (5 \rightharpoonup Author) \times Publisher$ $\cong_1 \qquad \{\ r_1 = id \rightharpoonup \langle\langle \pi_1, \pi_3\rangle, \pi_2\rangle\ ,\ f_1 = id \rightharpoonup \langle \pi_1 \cdot \pi_1, \pi_2, \pi_2 \cdot \pi_1\rangle\ \}$ $ISBN \rightharpoonup (Title \times Publisher) \times (5 \rightharpoonup Author)$ $\leq_2 \qquad \{$ (78): $r_2 = unnjoin\ ,\ f_2 = \bowtie_n\ \}$ $(ISBN \rightharpoonup Title \times Publisher) \times (ISBN \times 5 \rightharpoonup Author)$ $= \quad Books_2$ Sintetize as funções de abstracção e de representação globais, escrevendo-as em notação VDM-SL. □ *(v.s.f.f.)*

(cont.)

Exercício 14. Recorde a lei (30) que representa estruturas indutivas sob a forma de pares *(heap, apontador)*, bem como a respectiva função de abstracção (32). Seja $K \xleftarrow{f} K$ uma função de transformação de apontadores, usada como parâmtero na seguinte operação de re-alocação de células de um *heap* (vulg. *compressão*):

$$\begin{array}{rcl} compress & : & (K \longrightarrow K) \longrightarrow (K \rightharpoonup \mathsf{F}\, K) \longrightarrow (K \rightharpoonup \mathsf{F}\, K) \\ compress\ f\ M & \stackrel{\text{def}}{=} & (\mathsf{F}\, f) \cdot M \cdot f^\circ \end{array}$$

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 *compressão* $compress\ f\ M$ estará correcta se não destruir a representação, isto é, se

$$F(compress\ f\ M, f\ k) = F(M, k) \tag{33}$$

se verificar, onde F é a abstracção (32). Complete o cálculo seguinte de uma condição que é suficiente para (33) estar garantida. E que condição é essa?

$$\begin{array}{ll} & F(compress\ f M, f\ k) = F(M, k) \\ \equiv & \quad \{\ \ldots\ \} \\ & [\![(compress\ f M)]\!](f\ k) = [\![(M)]\!]k \\ \equiv & \quad \{\ \ldots\ \} \\ & [\![((\mathsf{F}\, f) \cdot M \cdot f^\circ)]\!] \cdot f = [\![(M)]\!] \\ \equiv & \quad \{\ \ldots\ \} \\ & [\![\ in, (\mathsf{F}\, f) \cdot M \cdot f^\circ\ ]\!] \cdot f = [\![\ in, M\ ]\!] \\ \Leftarrow & \quad \{\ \ldots\ \} \\ & (\mathsf{F}\, f) \cdot M \cdot f^\circ \cdot f = (\mathsf{F}\, f) \cdot M \\ \Leftarrow & \quad \{\ \ldots\ \} \\ & H \cdot f^\circ \cdot f = H \\ \equiv & \quad \{\ \ldots\ \} \\ & H \cdot f^\circ \cdot f \subseteq H \\ \Leftarrow & \quad \{\ \ldots\ \} \\ & f^\circ \cdot f \subseteq id \end{array}$$

□

O DOCENTE _____

AULA	SUMÁRIO
Teórica 2007.05.31 5.ª-feira, 14h00–16h00 Sala DI-A2 (LESI+LMCC)	Cálculo de ciclos `for/while` (126) a partir do hilomorfismo genérico (34) $$\begin{array}{lcl} f & : & A \longrightarrow C \\ f & = & p \rightarrow b\ ,\ \theta \cdot \langle d, f \cdot e \rangle \end{array} \quad (34)$$ Introdução de parâmetros de acumulação. Leis de factorização iterativa (132, 133). Uso do cálculo de hilomorfismos na prova de (132). O DOCENTE _____

AULA	SUMÁRIO

Prática
2007.06.04
2.ª-feira, 16h00–18h00
DI-A.2 (LMCC+LESI)
	Resolução dos exercícios seguintes:
	
	Exercício 15. É sabido que a compreensão de listas, eg. em VDM-SL
	`[ g(l(i)) | i in set inds l ]`
	é o catamorfismo
	$$(\!|in \cdot (id + g \times id)|\!)$$
	Mostre que esse catamorfismo é um ciclo `while` e escreva-o como tal em notação VDM com variáveis de estado interno. □
	
	Exercício 16. Verifique se a função f definida no fragmento de VDM-SL que se segue,
	`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;`
	está em condições de ser transformada num ciclo-while. Justifique adequadamente a sua resposta identificando eventuais lei de cálculo que tenha utilizado. □
	Considerações finais. Encerramento da disciplina. Preenchimento dos inquéritos de avaliação das aulas teóricas e práticas.
	
	O DOCENTE _____

Adenda aos sumários de MFP-II/0607

Contents

A Lecture notes

A.1 Introduction to reification by calculation

Calculation of (2) from (3), for $Spec = Post \cdot Pre$. First recall the two proof obligations at specification-level:

- *Satisfiability*:

$$Pre \quad \subseteq \quad \delta\, Post \tag{35}$$

- *Invariant preservation*:

$$Spec \cdot Inv \;\subseteq\; Inv \cdot Spec \tag{36}$$

Then we reason:

$$\begin{array}{ll}
 & Spec \vdash f \\
\equiv & \quad \{\ (3)\ \} \\
 & f \cdot \delta\, Spec \subseteq Spec \\
\equiv & \quad \{\ \text{substitution}\ \} \\
 & f \cdot \delta\,(Post \cdot Pre) \subseteq Post \cdot Pre \\
\equiv & \quad \{\ \text{domain of composition ; domain of coreflexive}\ \} \\
 & f \cdot (\delta\, Post) \cdot Pre \subseteq Post \cdot Pre \\
\equiv & \quad \{\ (157)\ \} \\
 & f \cdot (\delta\, Post) \cdot Pre \subseteq Post \\
\equiv & \quad \{\ \text{satisfiability}\ \} \\
 & f \cdot Pre \subseteq Post
\end{array}$$

B Case studies in VDM data refinement

B.1 *Hash Tables*

B.1.1 Abstract data type

Abstract data models

```
Collection = set of Data ;
Data     = int  ;
```
(37)

Abstract functionality Insert data into collection:

```
insert : Data -> Collection -> Collection
insert(d)(S) == S union {d} ;
```

Find data in collection:

```
belongs : Data -> Collection -> bool
belongs(d)(S) == d in set S;
```

B.1.2 Concrete data type

Concrete data models Define

```
HTable = map Location to set of Data
inv HT == forall k in set dom HT &
          HT(k) <> {} and forall d in set HT(k) & hash(d) = k;
Location = nat;
```
(38)

assuming some given hash function of type

```
hash : Data -> Location
```

Concrete functionality Implementing $insert$:

```
insertHT : Data -> HTable -> HTable
insertHT(d)(HT) == let k = hash(d)
        in HT ++ { k |-> (if k in set dom HT then HT(k) else {})
                         union {d}};
```

Implementing $belongs$:

```
belongsHT : Data -> HTable -> bool
belongsHT(d)(HT) == hash(d) in set dom HT and
                    d in set HT(hash(d));
```

B.1.3 Refinement step

Abstraction function

```
absf : HTable -> Collection
absf(HT) == dunion (rng HT) ;
```

Representation function

```
repf : Collection ->  HTable
repf(S) == { hash(x) |->  { d | d in set S & hash(d) = hash(x) } | x in set S };
```
(39)

B.1.4 Test data

```
S : Collection = { 1,13,2,34,5,10,26};
S1 : Collection = insert(45)(S);
HT : HTable = repf(S);
HTillegal : HTable = repf(S) ++ {1 |-> {1,13}};
HT1 : HTable = insertHT(45)(HT);
S1' : Collection = absf(HT1);
```

B.2 FS (file system) model

B.3 Abstract data type

B.3.1 Abstract data models

Hierarchical file systems are recursive data structures:

```
FS :: contents: map Id to Node; -- FS means file system

Node = File | FS;               -- a Node is either a file
                                -- or a directory

Id = token;                     -- node identifiers

File :: contents: FileContents;
FileContents = token;           -- file contents unspecified for the time being
```
(40)

Alternative (less abstract) data model — the *tar* (tape archive) file model:

```
Tar  = map Path to [FileContents] ; -- tar file
Path = seq of Id;                   -- file path
```
(41)

(This requires an invariant — guess which.) The representation function is a well-known (Linux/Unix) command:

```
tar : FS -> Tar
tar(mk_FS(M)) ==
   if   M = {|->}
   then {[] |-> nil}
   else merge (
               { cases M(k):
                   mk_File(f) -> { [k] |-> f } ,
                   others -> prefix(k,tar(M(k)))
                 end | k in set dom M });
```

B.4 Family tree model

B.5 Abstract data type

B.5.1 Abstract data model

```
GenDia :: indiv: Name
          mother: [GenDia]
          father: [GenDia];
Name   = seq of char;
```
(42)

B.5.2 Concrete data type

```
Heap :: heap: map K to (Name*[K]*[K])
        pointer: K
        inv mk_Heap(H,k) == k in set dom H and
                            let R = accessibility(H)
                            in closed(R) and wellf(R);
K = nat;
```
(43)

(Predicates `closed` and `wellf` left as exercise.)

B.5.3 Refinement step

Abstraction function. Calculating with law (30) yields

```
absf: Heap -> GenDia
absf(mk_Heap(H,k)) = f(H)(f);

f: (map K to (Name*[K]*[K])) -> K -> GenDia
f(H)(k) == mk_GenDia
             (H(k).#1,
              if H(k).#2 =nil then nil else f(H)(H(k).#2),
              if H(k).#3 =nil then nil else f(H)(H(k).#3)
             );
```
(44)

B.5.4 Test data

```
heapOk = {
        0 |-> mk_("Peter",1,2),
        1 |-> mk_("Mary",nil,4),
        4 |-> mk_("Jules",nil,nil),
        2 |-> mk_("Joseph",5,6),
        5 |-> mk_("Margaret",nil,nil),
        6 |-> mk_("Luigi",nil,nil)
        };

badHeap = {
        0 |-> mk_("Peter",1,2),
        1 |-> mk_("Mary",nil,4),
        4 |-> mk_("Jules",0  ,nil),
        2 |-> mk_("Joseph",5,6),
        5 |-> mk_("Margaret",nil,nil),
        6 |-> mk_("Luigi",nil,nil)
        };
```
(45)

C Catalogue of relational data refinement rules

In many cases, the abstraction/representation pair of a $\leq$ refinement rule is a pair of Galois-connected functions (f, r):

$$A \underset{f}{\overset{r}{\leq}} C \qquad (r\ a) \preceq b \equiv a \sqsubseteq (f\ b) \tag{46}$$

where the $\preceq, \sqsubseteq$ ordering are made explicit in each case. The most special case is that of isomorphisms, where these orderings are the identity relation:

$$A \underset{f}{\overset{r}{\cong}} C \qquad r\ a = b \equiv a = f\ b \tag{47}$$

that is

$$A \underset{f}{\overset{r}{\cong}} C \qquad r^\circ = f$$

C.1 Isomorphisms

Product

$$A \times B \underset{swap}{\overset{swap}{\cong}} B \times A \tag{48}$$

$$A \times (B \times C) \underset{\langle \pi_1 \cdot \pi_1, \langle \pi_2 \cdot \pi_1, \pi_2 \rangle \rangle}{\overset{}{\cong}} (A \times B) \times C \tag{49}$$

$$A \underset{\pi_1}{\overset{\langle id, ! \rangle}{\cong}} A \times 1 \tag{50}$$

$$A \underset{\pi_2}{\overset{\langle !, id \rangle}{\cong}} 1 \times A \tag{51}$$

etc

Coproduct

$$A + B \underset{coswap}{\overset{coswap}{\cong}} B + A \tag{52}$$

$$A + (B + C) \underset{[id + i_1\ , i_2 \cdot i_2]}{\overset{[i_1 \cdot i_1\ , i_2 + id]}{\cong}} (A + B) + C \tag{53}$$

$$A \underset{i_1^\circ}{\overset{i_1}{\cong}} A + 0 \tag{54}$$

$$A \underset{i_2^\circ}{\overset{i_2}{\underset{\cong}{\rightleftarrows}}} 0 + A \tag{55}$$

etc

Product and coproduct

$$C \times (A + B) \underset{distr}{\overset{undistr}{\underset{\cong}{\rightleftarrows}}} C \times A + C \times B \tag{56}$$

Exponentials (functions)

$$B^{C \times A} \underset{uncurry}{\overset{curry}{\underset{\cong}{\rightleftarrows}}} (B^A)^C \tag{57}$$

$$(B \times C)^A \underset{split}{\overset{\langle \pi_1 \cdot, \pi_2 \cdot \rangle}{\underset{\cong}{\rightleftarrows}}} B^A \times C^A \tag{58}$$

Relational *either* "Either" (where $A \to B$ denotes the set of all relations from A to B):

$$(B + C) \to A \underset{[\_\,,\_]}{\overset{[\_\,,\_]^\circ}{\underset{\cong}{\rightleftarrows}}} (B \to A) \times (C \to A) \tag{59}$$

This ismorphism is nothing but (150).

Power transpose Applicable to any relation:

$$A \to B \underset{(\in \cdot)}{\overset{\Lambda}{\underset{\cong}{\rightleftarrows}}} (\mathcal{P} B)^A \tag{60}$$

cf.

$$f = \Lambda R \quad \equiv \quad R = \in \cdot f \tag{61}$$

Corresponds to the view of relations as "set-valued" functions.

"Relational currying" Extension of *currying* to relations, as direct consequence of (61):

$$B \times C \to A \underset{}{\overset{\overline{\_}}{\underset{\cong}{\rightleftarrows}}} (C \to A)^B \tag{62}$$

where

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

The calculation of (62) follows:

$$\begin{array}{ll} & B \times C \to A \\ \cong & \quad \{\ \Lambda/(\in\cdot)\ \} \\ & (\mathcal{P}A)^{B\times C} \\ \cong & \quad \{\ curry/uncurry\ \} \\ & ((\mathcal{P}A)^{C})^{B} \\ \cong & \quad \{\ (\in\cdot)^{B}\ \} \\ & (C \to A)^{B} \end{array}$$

"Maybe" transpose Applicable to simple relations only [6]:

$$A \rightharpoonup B \underset{untot=(i_1^\circ\cdot)}{\overset{tot}{\cong}} (B+1)^A \tag{63}$$

that is,

$$f = tot\ M \quad \equiv \quad M = i_1^\circ \cdot f \tag{64}$$

Corresponds to the view of simple relations as "possibly-undefined" functions.

Sets are multisets By combining (63) with $\mathcal{P}A \cong 2^A$ we obtain

$$A \rightharpoonup 1 \underset{s2m}{\overset{dom}{\cong}} \mathcal{P}A \tag{65}$$

where $s2mS = ! \cdot \lceil S \rceil$ and dom is such that $\delta\ M = \lceil dom\ M \rceil$ (cf. the isomorphism between sets and coreflexives).

Viewing sets as coreflexives, (65) is nothing but

$$M = ! \cdot \Phi \quad \equiv \quad \delta\ M = \Phi \tag{66}$$

— a fact related to the definition ordering

$$R \preceq S \quad \equiv \quad \delta\ R \subseteq \delta\ S \quad \equiv \quad ! \cdot R \subseteq ! \cdot S \tag{67}$$

***Currying* simple relations**

$$B \times C \rightharpoonup A \underset{}{\overset{(\bar{\ })}{\cong}} (C \rightharpoonup A)^B \tag{68}$$

This is the instance of (62) for simple relations. It will be referred to as the **multiple-key** decomposition / synthesis isomorphism.

C.2 "Less than" rules

C.2.1 Galois connections

Relational "split"

$$A \to B \times C \underset{\bowtie}{\overset{unjoin}{\leq}} (A \to B) \times (A \to C) \tag{69}$$

where R, S are arbitrary relations and the *join* operator is nothing but the *split* combinator:

$$R \bowtie S \quad \stackrel{\text{def}}{=} \quad \langle R, S \rangle$$

Exercise: guess $unjoin$. The GC of this rule is universal-property (149).

Simple power-transpose

$$A \to B \;\underset{(\in\cdot)}{\overset{collect}{\leq}}\; A \rightharpoonup \mathcal{P}B \tag{70}$$

where

$$collect\ R \;\stackrel{\text{def}}{=}\; (\Lambda R)\cdot \delta\, R \qquad \text{cf.} \qquad \begin{array}{ccccc} B & \xleftarrow{R} & A & \xleftarrow{\delta\, R} & A \\ {\scriptstyle\in}\uparrow & \swarrow_{f=\Lambda R} & & \swarrow_{collect\ R} & \\ \mathcal{P}B & & & & \end{array} \tag{71}$$

This corresponds to transposing relations into powerset-valued *simple* relations instead of (entire) functions, such as in (61). The GC captured by the diagram above is

$$collect\ R \mathrel{\dot\sqsubseteq} M \;\equiv\; R \subseteq \in \cdot M \tag{72}$$

where M is simple and $\dot\sqsubseteq$ is defined by

$$M \mathrel{\dot\sqsubseteq} N \;\equiv\; M \subseteq (\in \setminus \in)\cdot N \tag{73}$$

which expands (pointwise) to

$$M \mathrel{\dot\sqsubseteq} N \;\equiv\; \langle \forall\, a \in\ dom\ M\ ::\ a \in\ dom\ N\ \wedge\ \langle \forall\, b\ :\ b \in\ M\ a :\ b \in N\ a \rangle\rangle \tag{74}$$

The following proof of GC (70) is carried out by cyclic implication:

$$\begin{array}{ll} & collect\ R \mathrel{\dot\sqsubseteq} M \\ \Rightarrow & \quad \{\ \text{definition of } \dot\sqsubseteq \text{; monotonicity (GC)}\ \} \\ & \in \cdot\ collect\ R \subseteq \in \cdot (\in \setminus \in)\cdot M \\ \Rightarrow & \quad \{\ \in\cdot(collect\ R) = R \text{; } (\in\cdot) \text{ cancellation}\ \} \\ & R \subseteq \in \cdot M \\ \Rightarrow & \quad \{\ \text{monotonicity (GC) again}\ \} \\ & R \cdot M^\circ \subseteq \in \cdot M \cdot M^\circ \\ \Rightarrow & \quad \{\ R\cdot(\delta\, R) = R \text{; } M \text{ is simple ; transitivity of } \Rightarrow\ \} \\ & R \cdot (\delta\, R)\cdot M^\circ \subseteq \in \\ \equiv & \quad \{\ \Lambda\text{-cancellation (24)}\ \} \\ & \in \cdot (\Lambda R)\cdot(\delta\, R)\cdot M^\circ \subseteq \in \\ \equiv & \quad \{\ \text{shunting over GC of left division}\ \} \\ & (\Lambda R)\cdot(\delta\, R)\cdot M^\circ \subseteq \in \setminus \in \\ \equiv & \quad \{\ \text{“almost GC” (152)}\ \} \\ & (\Lambda R)\cdot(\delta\, R)\cdot(\delta\, M) \subseteq (\in \setminus \in)\cdot M \\ \equiv & \quad \{\ R \subseteq \in\cdot M \text{ above implies } \delta\, R \subseteq \delta\, M\ \} \\ & (\Lambda R)\cdot(\delta\, R) \subseteq (\in \setminus \in)\cdot M \\ \equiv & \quad \{\ \text{definitions}\ \} \\ & collect\ R \mathrel{\dot\sqsubseteq} M \end{array}$$

"Currying /uncurrying" among simple relations

$$(B \times C) \rightharpoonup A \;\underset{unpcurry}{\overset{pcurry}{\leftrightarrows}}\; B \rightharpoonup (C \rightharpoonup A) \qquad \leq \tag{75}$$

This is the *simple* counterpart of *relational currying* (68), therefore similar to what has just been reckoned about $collect$. Thus the definition which follows:

$$pcurry\ M \quad \stackrel{\text{def}}{=} \quad (id - \mathsf{img}\ \underline{\bot}) \cdot (\overline{M}) \tag{76}$$

and GC

$$pcurry\ M \stackrel{\cdot}{\subseteq} N \quad \equiv \quad M \subseteq unpcurry\ N \tag{77}$$

Exercise: guess $unpcurry$.

"Nested join"

$$A \rightharpoonup (D \times (B \rightharpoonup C)) \;\underset{\bowtie_n}{\overset{unnjoin}{\leftrightarrows}}\; (A \rightharpoonup D) \times ((A \times B) \rightharpoonup C) \qquad \leq \tag{78}$$

where

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

Concrete invariant induced by $unnjoin$:

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

C.2.2 Others

"Coproduct join"

$$A \rightharpoonup (B + C) \;\underset{\overset{+}{\bowtie}}{\overset{uncojoin}{\leftrightarrows}}\; (A \rightharpoonup B) \times (A \rightharpoonup C) \qquad \leq \tag{79}$$

$$M \overset{+}{\bowtie} N \quad \stackrel{\text{def}}{=} \quad (i_1 \cdot M) \cup (i_2 \cdot N) \tag{80}$$

Note that $\overset{+}{\bowtie}$ is not entire, because the union of two simple relations is not always simple. Let us thus calculate its (weakest) pre-condition:

$$\begin{array}{ll} & M \overset{+}{\bowtie} N \text{ is simple} \\ \equiv & \quad \{ \text{ definition } \} \\ & (i_1 \cdot M) \cup (i_2 \cdot N) \text{ is simple} \\ \equiv & \quad \{ \text{ simplicity of union of simple relations } \} \\ & (i_1 \cdot M) \cdot (i_2 \cdot N)^\circ \subseteq id \\ \equiv & \quad \{ \text{ converses ; shunting etc } \} \\ & M \cdot N^\circ \subseteq i_1^\circ \cdot i_2 \\ \equiv & \quad \{ \ i_1^\circ \cdot i_2 = \bot \text{ ; (151,152) } \} \\ & \delta\, M \cdot \delta\, N \subseteq \bot \\ \equiv & \quad \{ \text{ coreflexives } \} \\ & \delta\, M \cap \delta\, N \subseteq \bot \end{array}$$

Thus, the domains of M and N must be disjoint.

Handling lists Several other $\leq$ laws, eg.

$$A^\star \underset{list}{\overset{seq2index}{\underset{\leq}{\rightleftarrows}}} \mathbb{N} \rightharpoonup A$$

such that, for instance,

$$\begin{array}{rcl} seq2index\ [a,b,a] & = & \{(a,1),(b,2),(a,3)\} \\ list\ \{(a,11),(b,12),(a,33)\} & = & [a,b,a] \end{array}$$

Guess $list$ and $seq2index$.

C.3 Properties of the $\leq$-ordering

$\leq$ is a preorder Reflexivity is immediate

$$A \underset{id}{\overset{id}{\underset{\leq}{\rightleftarrows}}} A \quad \text{cf.} \quad id \cdot id = id \tag{81}$$

while transitivity

$$A \underset{F}{\overset{R}{\underset{\leq}{\rightleftarrows}}} B \ \wedge\ B \underset{G}{\overset{S}{\underset{\leq}{\rightleftarrows}}} C \ \Rightarrow\ A \underset{F\cdot G}{\overset{S\cdot R}{\underset{\leq}{\rightleftarrows}}} C \tag{82}$$

is calculated as follows:

- First note that, by monotonicity, the composite representation is *connected* to the composite abstraction:

$$S \cdot R \subseteq (F \cdot G)^\circ \tag{83}$$

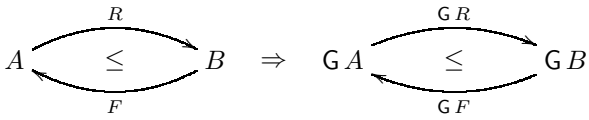
- The show show that composite abstraction is simple and surjective:

$$\begin{array}{ll} & \mathsf{img}\,(F \cdot G) = id \\ \equiv & \quad \{ \text{ expanding and converses } \} \\ & F \cdot (\mathsf{img}\, G) \cdot F^\circ = id \\ \equiv & \quad \{ G \text{ is simple and surjective } \} \\ & \mathsf{img}\, F = id \\ \equiv & \quad \{ F \text{ is simple and surjective } \} \\ & id = id \end{array}$$

- Finally, show that composite representation is injective and entire:

$$\begin{array}{ll} & \mathsf{ker}\,(S \cdot R) = id \\ \equiv & \quad \{ \text{ expanding and converses } \} \\ & R^\circ \cdot (\mathsf{ker}\, S) \cdot R = id \\ \equiv & \quad \{ S \text{ is injective and entire } \} \\ & \mathsf{ker}\, R = id \\ \equiv & \quad \{ R \text{ is injective and entire } \} \\ & id = id \end{array}$$

Structural data refinement Let G be a parametric datatype. Then



(84)

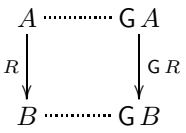
provided G has the properties of a *relator* — see section D.

It is easy to prove that $\mathsf{G}\,F$ is an abstraction and that $\mathsf{G}\,R$ is a representation wherever F and R are so, respectively. The proof that they are connected goes as follows:

$$
\begin{array}{ll}
 & \mathsf{G}\,R \subseteq (\mathsf{G}\,F)^\circ \\
\equiv & \quad \{ \text{ relators commute with converse (88) } \} \\
 & \mathsf{G}\,R \subseteq \mathsf{G}(F^\circ) \\
\Leftarrow & \quad \{ \text{ relators are monotonic (89) } \} \\
 & R \subseteq F^\circ
\end{array}
$$

D Relators

Relators [2] have to do with parametric datatyping: a parametric datatype G is said to be a relator wherever, given a relation from A to B, $\mathsf{G}\,R$ extends R to G-structures: it is a relation from $\mathsf{G}\,A$ to $\mathsf{G}\,B$



(85)

which obeys the following properties: it commutes with the identity, with composition and with converse,

$$\mathsf{G}\,id = id \tag{86}$$

$$\mathsf{G}\,(R \cdot S) = (\mathsf{G}\,R) \cdot (\mathsf{G}\,S) \tag{87}$$

$$\mathsf{G}\,(R^\circ) = (\mathsf{G}\,R)^\circ \tag{88}$$

and is monotonic:

$$R \subseteq S \;\Rightarrow\; \mathsf{G}\,R \subseteq \mathsf{G}\,S \tag{89}$$

All in all, a relator is the extension of a *functor* from functions to relations. In fact, once R, S above are restricted to functions, (88) and (89) become trivialities — the former establishing that G preserves isomorphisms and the latter that G preserves equality (Leibniz).

It is easy to show that relators preserve all basic properties of relations. Let us calculate the simplicity of $\mathsf{G}\,R$, for instance:

$$
\begin{array}{ll}
 & (\mathsf{G}\,R) \cdot (\mathsf{G}\,R)^\circ \subseteq id \\
\equiv & \quad \{ \text{ (88) } \} \\
 & (\mathsf{G}\,R) \cdot (\mathsf{G}\,R^\circ) \subseteq id \\
\equiv & \quad \{ \text{ (87) and (86) } \} \\
 & \mathsf{G}(R \cdot R^\circ) \subseteq \mathsf{G}\,id \\
\Leftarrow & \quad \{ \text{ (89) } \} \\
 & R \cdot R^\circ \subseteq id
\end{array}
$$

So, if R is simple, $\mathsf{G}\,R$ will be simple too.

Examples The most simple relators are the *identity* relator Id, which is such that

$$\begin{aligned} \mathsf{Id}\ A &= A \\ \mathsf{Id}\ R &= R \end{aligned}$$

and the *constant* relator K (for a particular concrete data type K) which is such that

$$\begin{aligned} \mathsf{K}\ A &= K \\ \mathsf{K}\ R &= id_K \end{aligned}$$

List maps extend to relators in the obvious way,

$$l(R^\star)l' \quad \equiv \quad \mathsf{length}\, l = \mathsf{length}\, l' \ \wedge\ \forall i \in \mathsf{inds}\, l.(l\ i)R(l'\ i) \tag{90}$$

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

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2 \rangle \tag{91}$$

$$R + S = [i_1 \cdot R\ , i_2 \cdot S] \tag{92}$$

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 (168) and (175), 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 made available in VDM and Haskell. For instance, the $Maybe$ datatype is an implementation of polynomial relator $\mathsf{G} = \mathsf{Id} + 1$ (ie. $\mathsf{G}\ A = A + 1$), where 1 denotes the *singleton* datatype.

The $(A \rightharpoonup\ )$ relator The prominent rôle of the parametric type $\mathsf{G}\ X = A \rightharpoonup X$, for A a given datatype A of *keys*, leads us to the investigation of its properties as a relator,

$$\begin{array}{ccc} B & \cdots\cdots & A \rightharpoonup B \\ {\scriptstyle R}\downarrow & & \downarrow{\scriptstyle A \rightharpoonup R} \\ C & \cdots\cdots & A \rightharpoonup C \end{array} \tag{93}$$

where we define $A \rightharpoonup R$ as follows:

$$N(A \rightharpoonup R)M \quad \stackrel{\text{def}}{=} \quad \delta\, M = \delta\, N \ \wedge\ N \cdot M^\circ \subseteq R \tag{94}$$

So, wherever N and M are $A \rightharpoonup R$-related, they are equally defined and their outputs are R-related, that is, in VDM

```
dom M = dom N and forall k in set dom M & post_R(N(k),M(k))
```

Wherever R is a function f, then $A \rightharpoonup f$ is also a function, defined by projection

$$(A \rightharpoonup f)M = f \cdot M \tag{95}$$

This can be extended to a bi-relator,

$$(g \rightharpoonup f)M = f \cdot M \cdot g^\circ \tag{96}$$

provided g is injective (thus ensuring the simplicity of $(g \rightharpoonup f)M$).

Membership. Equation (22) involves the set-theoretic membership relation $A \xleftarrow{\in} \mathcal{P}A$. Sentence $a \in x$ (meaning that "a belongs to x" or "a occurs in x") can be generalized to x's other than sets. For instance, one may check whether a particular integer occurs in one or more leaves of a binary tree, or of any other *collective* or *container* type F.

Such a generic membership relation will have type $A \xleftarrow{\in} \mathsf{F}\ A$, where F is a type *parametric on* A. Technically, the parametricity of F is captured by regarding it as a *relator* — as we have seen above.

There is more than one way to generalize $A \xleftarrow{\in} \mathcal{P}A$ to relators other than the powerset. (For a thorough presentation of the subject see chapter 4 of [3].) For our purposes it will be enough to say that $A \xleftarrow{\in_{\mathsf{F}}} \mathsf{F}\,A$, if it exists, is a *lax natural transformation*, that is,

$$\in_{\mathsf{F}} \cdot \mathsf{F}\,R \quad \subseteq \quad R \cdot \in_{\mathsf{F}} \tag{97}$$

$$\begin{array}{ccc} A & \xleftarrow{\in_{\mathsf{F}}} & \mathsf{F}\,A \\ {\scriptstyle R}\downarrow & \supseteq & \downarrow{\scriptstyle \mathsf{F}\,R} \\ B & \xleftarrow[\in_{\mathsf{F}}]{} & \mathsf{F}\,B \end{array}$$

holds. This is an obvious requirement, as can be checked by the following example: we convert (97) to pointfree notation for *lists* of natural numbers ordered by "smaller than":

$$\langle \exists\, l \;::\; l{\leq^\star}l' \;\wedge\; b \in elems\, l \rangle \quad \Rightarrow \quad \langle \exists\, a \;:\; a \in elems\, l' :\; b \leq a \rangle$$

The membership associated to *polynomial* relators is defined inductively over the structure of such relators:

- Constant and identity functors:

$$\begin{array}{rcl} \in_C & \stackrel{\text{def}}{=} & \bot \\ \in_{\lambda X.X} & \stackrel{\text{def}}{=} & id \end{array}$$

- Product and coproduct

$$\begin{array}{rcl} \in_{\mathsf{F}\times\mathsf{G}} & \stackrel{\text{def}}{=} & (\in_{\mathsf{F}} \cdot \pi_1) \cup (\in_{\mathsf{G}} \cdot \pi_2) \\ \in_{\mathsf{F}+\mathsf{G}} & \stackrel{\text{def}}{=} & [\in_{\mathsf{F}}\,, \in_{\mathsf{G}}] \end{array}$$

- Composition

$$\in_{\mathsf{F}\cdot\mathsf{G}} \quad \stackrel{\text{def}}{=} \quad \in_{\mathsf{G}} \cdot \in_{\mathsf{F}}$$

Type functors also have membership. Here is an example:

$$\in_{X^\star} \quad \stackrel{\text{def}}{=} \quad \in \cdot\, elems$$

Examples Let $\mathsf{F}\,X = K + K' \times X^\star$ in

$$\begin{array}{rl} & \in_{K+K'\times X^\star} \\ = & \quad \{ \;\in \text{ for coproduct bifunctor } \} \\ & [\in_K\,, \in_{K'\times X^\star}] \\ = & \quad \{ \;\in \text{ for constant and product (bi)functors } \} \\ & [\bot\,, (\in_{K'} \cdot \pi_1) \cup (\in_{X^\star} \cdot \pi_2)] \\ = & \quad \{ \;\in \text{ for constant and identity functor } \} \\ & [\bot\,, (\bot \cdot \pi_1) \cup (\in \cdot\, elems \cdot \pi_2)] \\ = & \quad \{ \;\bot \text{ and } [R\,, S] = (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \;\} \\ & \in \cdot\, elems \cdot \pi_2 \cdot i_2^\circ \end{array}$$

Going pointwise:

$$\begin{array}{rl} & k \in_{K+K'\times X^\star} x \\ \equiv & \quad \{ \text{ calculation above } \} \\ & k(\in \cdot\, elems \cdot \pi_2 \cdot i_2^\circ)x \\ \equiv & \quad \{ \text{ relational composition } \} \\ & k(\in \cdot\, elems \cdot \pi_2)(a,l) \;\wedge\; x = i_2(a,l) \\ \equiv & \quad \{ \text{ trivia } \} \\ & k \in (elems\, l) \;\wedge\; x = i_2(a,l) \end{array}$$

Another example: Let $\mathsf{F}\,X = 1 + A \times X$. Then,

$$
\begin{array}{ll}
 & \in_{1+A\times X} \\
= & \quad \{ \ \in \text{ for coproduct bifunctor } \} \\
 & [\in_1 , \in_{A\times X}] \\
= & \quad \{ \ \in \text{ for constant and product (bi)functors } \} \\
 & [\bot , (\in_A \cdot \pi_1) \cup (\in_{\lambda X.X} \cdot \pi_2)] \\
= & \quad \{ \ \in \text{ for constant and identity functor } \} \\
 & [\bot , (\bot \cdot \pi_1) \cup (id \cdot \pi_2)] \\
= & \quad \{ \ \bot \text{ and } [R , S] = (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \ \} \\
 & \pi_2 \cdot i_2^\circ
\end{array}
$$

Going pointwise:

$$
\begin{array}{ll}
 & k \in_{1+A\times X} x \\
\equiv & \quad \{ \text{ calculation above } \} \\
 & k(\pi_2 \cdot i_2^\circ)x \\
\equiv & \quad \{ \text{ relational composition } \} \\
 & k(\pi_2)(a, k') \ \wedge \ x = i_2(a, k') \\
\equiv & \quad \{ \text{ trivia } \} \\
 & x = i_2(a, k') \ \wedge \ k = k' \\
\equiv & \quad \{ \text{ trivia } \} \\
 & x = i_2(a, k)
\end{array}
$$

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

$$T \cdot \mathsf{F}\,R \quad \subseteq \quad \mathsf{G}\,R \cdot T \tag{98}$$

holds, cf. (omitting subscripts):

$$
\begin{array}{ccc}
A & \mathsf{G}\,A \xleftarrow{\ T\ } & \mathsf{F}\,A \\
{\scriptstyle R}\downarrow & {\scriptstyle \mathsf{G}\,R}\downarrow \quad \supseteq & \downarrow{\scriptstyle \mathsf{F}\,R} \\
B & \mathsf{G}\,B \xleftarrow[\ T\ ]{} & \mathsf{F}\,B
\end{array}
$$

E Examples of data refinement by calculation

E.1 The BAMS example

$$
\begin{array}{ll}
 & AccId \rightharpoonup (\mathcal{P} AccHolder \times Amount) \\
\cong_1 & \quad \{ \text{ (65) } \} \\
 & AccId \rightharpoonup ((AccHolder \rightharpoonup 1) \times Amount) \\
\cong_2 & \quad \{ \text{ product swapping } \} \\
 & AccId \rightharpoonup (Amount \times (AccHolder \rightharpoonup 1)) \\
\leq_3 & \quad \{ \text{ (78) } \} \\
 & (AccId \rightharpoonup Amount) \times ((AccId \times AccHolder) \rightharpoonup 1) \\
\cong_4 & \quad \{ \text{ (65) } \} \\
 & (AccId \rightharpoonup Amount) \times \mathcal{P}(AccId \times AccHolder)
\end{array}
$$

Abstractions and representations:

$$
\begin{array}{rcl}
\cong_1 & = & \left\{ \begin{array}{l} f = id \rightharpoonup (dom \times id) \\ r = id \rightharpoonup (set2fm \times id) \end{array} \right. \\
\cong_2 & = & \left\{ \begin{array}{l} f = id \rightharpoonup swap \\ r = id \rightharpoonup swap \end{array} \right. \\
\leq_3 & = & \left\{ \begin{array}{l} f = \bowtie_n \\ r = unnjoin \end{array} \right. \\
\cong_4 & = & \left\{ \begin{array}{l} f = id \times set2fm \\ r = id \times dom \end{array} \right.
\end{array}
$$

E.2 The PPD example

Production planning department database:

```
PPD :: comps:  CompDb
       equips: EquipDb
    inv(mk_PPD(C,E)) == okRefIntegrity(C,E);
```
(99)

Components:

```
CompDb     = map CompId to Comp;
Comp      :: stock: StockInfo
             cost:  real;
CompId    = token;
StockInfo :: alarm: nat
             instock: nat
             description: seq of char;
```
(100)

Equipments:

```
EquipDb    = map EquipId to Equip;
EquipId    = token;
Equip     :: stock: StockInfo
             ptree: Unitbag;
Unitbag    = map Unit to nat;
Unit    = EquipId' | CompId';
EquipId' :: key: EquipId;
CompId'  :: key: CompId;
```
(101)

Data model in calculational style:

$$\begin{array}{rcl} PPD & = & CompDb \times EquipDb \\ CompDb & = & CompId \rightharpoonup S \times \mathbb{R} \\ EquipDb & = & EquipId \rightharpoonup (S \times (Unit \rightharpoonup \mathbb{N})) \\ Unit & = & CompId + EquipId \end{array}$$

Calculation steps:

$$\begin{array}{ll} & PPD \\ = & \quad \{ \text{ data definitions } \} \\ & (CompId \rightharpoonup S \times \mathbb{R}) \times \\ & (EquipId \rightharpoonup (S \times (CompId + EquipId) \rightharpoonup \mathbb{N})) \\ \\ \leq_1 & \quad \{ \text{ law (78) in } \left\{ \begin{array}{l} r_1 = id \times unnjoin \\ f_1 = id \times \bowtie_n \end{array} \right. \} \\ \\ & (CompId \rightharpoonup S \times \mathbb{R}) \times \\ & ((EquipId \rightharpoonup S) \times ((EquipId \times (CompId + EquipId)) \rightharpoonup \mathbb{N})) \\ \\ \cong_2 & \quad \{ \text{ products, etc in } \left\{ \begin{array}{l} r_2 = assocl \\ f_2 = assocr \end{array} \right. \} \\ \\ & ((CompId \rightharpoonup S \times \mathbb{R}) \times (EquipId \rightharpoonup S)) \times \\ & ((EquipId \times (CompId + EquipId)) \rightharpoonup \mathbb{N}) \\ \\ \cong_3 & \quad \{ \text{ products etc. in } \left\{ \begin{array}{l} r_3 = id \times (distr \rightharpoonup id) \\ f_3 = id \times (undistr \rightharpoonup id) \end{array} \right. \} \\ \\ & ((CompId \rightharpoonup S \times \mathbb{R}) \times (EquipId \rightharpoonup S)) \times \\ & (((EquipId \times CompId) + (EquipId \times EquipId)) \rightharpoonup \mathbb{N}) \\ \\ \cong_4 & \quad \{ \text{ law (79) } \left\{ \begin{array}{l} r_4 = id \times (\overset{+}{\bowtie})^\circ \\ f_4 = id \times \overset{+}{\bowtie} \end{array} \right. \} \\ \\ & ((CompId \rightharpoonup (N \times N \times D) \times \mathbb{R}) \times (EquipId \rightharpoonup (N \times N \times D))) \times \\ & (((EquipId \times CompId) \rightharpoonup \mathbb{N}) \times ((EquipId \times EquipId) \rightharpoonup \mathbb{N})) \\ \\ \cong_5 & \quad \{ \text{ guess } flatten \text{ and its converse in } \left\{ \begin{array}{l} r_5 = (id \rightharpoonup flatten \times id) \times id \\ f_5 = (id \rightharpoonup flatten^\circ \times id) \times id \end{array} \right. \} \\ \\ & ((CompId \rightharpoonup N \times N \times D \times \mathbb{R}) \times (EquipId \rightharpoonup N \times N \times D)) \times \\ & (((EquipId \times CompId) \rightharpoonup \mathbb{N}) \times ((EquipId \times EquipId) \rightharpoonup \mathbb{N})) \\ \\ \cong_6 & \quad \{ \text{ calculate } f_6 \text{ in } \left\{ \begin{array}{l} r_6 = \langle \pi_1 \cdot \pi_1, \pi_2 \cdot \pi_1, \pi_1 \cdot \pi_2, \pi_2 \cdot \pi_2 \rangle \\ f_6 = r_6^\circ \end{array} \right. \} \\ \\ & (CompId \rightharpoonup \mathbb{N} \times \mathbb{N} \times D \times \mathbb{R}) \times \end{array}$$

$$
\begin{aligned}
&(EquipId \rightharpoonup \mathbb{N} \times \mathbb{N} \times D) \times \\
&((EquipId \times CompId) \rightharpoonup \mathbb{N}) \times \\
&((EquipId \times EquipId) \rightharpoonup \mathbb{N})
\end{aligned}
$$

SQL encoding (after deciding about primitive datatypes, eg. description, etc):

```
CREATE TABLE COMPONENTS (
       CompId CHAR   (8)   NOT NULL,
       CAlarm NUMERIC  (10)  NOT NULL,
       CStock NUMERIC (10)  NOT NULL,
       CDescription CHAR (73) NOT NULL,
       Cost NUMERIC   (6,3) NOT NULL
       CONSTRAINT COMPONENTS_pk PRIMARY KEY(CompId)
       );

CREATE TABLE EQUIPMENTS (
       EquipId CHAR (8) NOT NULL,
       CAlarm NUMERIC  (10)  NOT NULL,
       EStock NUMERIC (10) NOT NULL,
       EDescription CHAR (73) NOT NULL,
       CONSTRAINT EQUIPMENTS_pk PRIMARY KEY (EquipId)
       );

CREATE TABLE PART_OF (
       Comp CHAR (8) NOT NULL,
       Equip CHAR (8) NOT NULL,
       HowManyC NUMERIC (10) NOT NULL,
       CONSTRAINT PART_OF_pk PRIMARY KEY (Comp,Equip)
       );

CREATE TABLE SUBEQ_OF (
       EquipA CHAR (8) NOT NULL,
       EquipB CHAR (8) NOT NULL,
       HowManyE NUMERIC (10) NOT NULL,
       CONSTRAINT PART_OF_pk PRIMARY KEY (EquipA,EquipB)
       );

ALTER TABLE PART_OF ADD CONSTRAINT PART_OF_fk1
      FOREIGN KEY (Comp) REFERENCES COMPONENTS(CompId);

ALTER TABLE PART_OF ADD CONSTRAINT PART_OF_fk2 FOREIGN KEY (Equip)
      REFERENCES EQUIPMENTS(EquipId);

ALTER TABLE SUBEQ_OF ADD CONSTRAINT SUBEQ_OF_fk1 FOREIGN KEY (EquipA)
      REFERENCES EQUIPMENTS(EquipId);

ALTER TABLE SUBEQ_OF ADD CONSTRAINT SUBEQ_OF_fk2 FOREIGN KEY (EquipB)
      REFERENCES EQUIPMENTS(EquipId);
```

F Elements of the Fixpoint Calculus

F.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*

$$a \quad \leq_A f\, a \tag{102}$$

is said to be a post-fixpoint *of f.*

- *every $a \in A$ such that*

$$a \quad \geq_A f\, a \tag{103}$$

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*

$$a \quad = \quad f\, a \tag{104}$$

holds.

□

Examples:

- Given endofunction

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

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

- Let $P \xleftarrow{R} P$ be a relation on nonempty P in

$$x = R \cup R \cdot x \tag{105}$$

Define

$$f\, x \quad \stackrel{\text{def}}{=} \quad R \cup R \cdot x \tag{106}$$

on the poset of all P to P relations, ordered by $\subseteq$. Then

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

Clearly, every fixpoint $a = f\, a$ can be regarded as a "solution" to equation

$$x = f\, x \tag{107}$$

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!

F.2 Hylomorphisms are least fixpoints

Relational hylomorphisms:

$$\begin{array}{ccc} A & \xrightarrow{S} & \mathsf{F}\,A \\ {\scriptstyle X}\downarrow & & \downarrow{\scriptstyle \mathsf{F}\,X} \\ B & \xleftarrow[R]{} & \mathsf{F}\,B \end{array} \quad \text{that is,} \quad X = R \cdot (\mathsf{F}\,X) \cdot S$$

F.2.1 Examples: VDM collective types

Browsing the `set of..` collecting type:

$$\begin{array}{ccc} \texttt{set of}\ A & \xrightarrow{Ins^\circ} & 1 + A \times \texttt{set of}\ A \\ {\scriptstyle \{\!|R|\!\}}\downarrow & & \downarrow{\scriptstyle id + id \times \{\!|R|\!\}} \\ B & \xleftarrow[R]{} & 1 + A \times B \end{array}$$

Let us introduce notation

$$\{\!|R|\!\} \quad = \quad [\![\, R, Ins^\circ \,]\!]$$

where

$$Ins \quad \stackrel{\text{def}}{=} \quad [\underline{\emptyset}\,, Puts]$$

and (in VDM-SL)

```
Puts[@A] :  @A * set of @A -> set of @A
Puts(e,s) == {e} union s
pre not e in set s ;
```

Pointwise (parametric VDM-SL) version of $\{\!|R|\!\}$, for $R = [\underline{u}\,, f]$:

```
shylo[@A,@B] : (@A*@B -> @B) * @B -> set of @A -> @B
shylo(f,u)(s) ==
      if s={} then u
      else let a in set s,
               r = s \ {a}
           in f(a,shylo[@A,@B](f,u)(r));
```

Browsing the `map..to..` collective type:

$$\begin{array}{ccc} \texttt{map}\ A\ \texttt{to}\ C & \xrightarrow{Ins^\circ} & 1 + (A \times C) \times \texttt{map}\ A\ \texttt{to}\ C \\ {\scriptstyle \{\!|R|\!\}}\downarrow & & \downarrow{\scriptstyle id + id \times \langle id, \{\!|R|\!\}\rangle} \\ B & \xleftarrow[R]{} & 1 + (A \times C) \times B \end{array}$$

leading to the following pointwise syntax:

```
mhylo[@A,@B,@C] : (@A*@C*@B -> @B) * @B ->
                  map @A to @C ->
                  @B
mhylo(f,u)(M) ==
      if M={|->} then u
      else let a in set dom M,
               c = M(a),
               R = {a} <-: M
           in f(c,mhylo[@A,@B,@C](f,u)(R));
```

F.2.2 Examples: relational ana's and cata's

We define

$$\begin{array}{lcl} (\!|R|\!) & = & [\![R, in^\circ]\!] \\ [\!(S)\!] & = & [\![in, S]\!] \end{array}$$

where $\mu \mathsf{F} \underset{in}{\overset{in^\circ}{\cong}} \mathsf{F}\,\mu\mathsf{F}$. For instance,

$$elems = (\!|ins|\!)$$

Question: how do we *reason* about relational hylos?

F.3 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*

$$f \cdot \leq_A \quad \subseteq \quad \leq_B \cdot f \tag{108}$$

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 $(A; \leq)$*;*
- P *be the set of all fixpoints of* f*, i.é*

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

Then

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

$$\bigwedge P \quad = \quad \bigwedge \{x \mid x \geq f\,x\} \tag{109}$$

$$\bigvee P \quad = \quad \bigvee \{x \mid x \leq f\,x\} \tag{110}$$

We define:

$$\mu f \quad \stackrel{\text{def}}{=} \quad \bigwedge P \tag{111}$$

$$\nu f \quad \stackrel{\text{def}}{=} \quad \bigvee P \tag{112}$$

□

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

F.4 Laws of the Fixpoint Calculus

F.4.1 Computation rule:

$$\mu f \quad = \quad f\,\mu f \tag{113}$$

Example: hylo-cancellation law

$$[\![\, R, S \,]\!] \quad = \quad R \cdot \mathsf{F}\,[\![\, R, S \,]\!] \cdot S$$

F.4.2 Rolling rule:

$$\mu(g \cdot f) \quad = \quad g(\mu(f \cdot g)) \tag{114}$$

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

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

that is,

$$[\![\, R, S \,]\!] \quad = \quad R \cdot [\![\, \mathsf{F}\,R, S \,]\!]$$

F.4.3 Square rule:

$$\mu f = \mu(f^2) \tag{115}$$

F.4.4 Monotonicity:

$$\mu f \leq \mu g \quad \Leftarrow \quad f \dot{\leq} g \tag{116}$$

where $\dot{\leq}$ is defined by (145).

F.4.5 Induction rule:

$$\mu f \leq x \quad \Leftarrow \quad f\,x \leq x \tag{117}$$

F.4.6 Diagonal rule:

given monotonic $x\;\theta\;y$

$$\langle \mu\, x \;::\; \langle \mu\, y \;::\; x\;\theta\;y \rangle \rangle \quad = \quad \langle \mu\, x \;::\; x\;\theta\;x \rangle \tag{118}$$

F.4.7 μ-fusion rule:

The μ-fusion theorem which follows is perhaps the most useful rule of the fixpoint calculus:

Theorem 2 *(μ-fusion theorem) Let*

$$\begin{array}{ccc} A & \xleftarrow{f^\flat} & B \\ {\scriptstyle g}\uparrow & & \uparrow{\scriptstyle h} \\ A & \xleftarrow[f^\flat]{} & B \end{array}$$

- h, g *be monotonic,*
- $(A, \leq)$ *and* $(B, \sqsubseteq)$ *be complete* ***lattices****,*
- $f^\flat$ *be a lower-adjoint in a Galois connection.*

Then

$$f^\flat(\mu h) = \mu g \quad \Leftarrow \quad f^\flat \cdot h = g \cdot f^\flat \tag{119}$$

F.4.8 Applications of μ-fusion theorem

Converse of hylo

$$[\![S, R]\!]^\circ = [\![R^\circ, S^\circ]\!] \tag{120}$$

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

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

that is,

$$\begin{array}{l} \mu h = [\![S, R]\!] \\ \mu g = [\![T, U]\!] \end{array}$$

Then

$$\begin{array}{ll} & [\![S, R]\!]^\circ = [\![T, U]\!] \\ \Leftarrow & \quad \{\ \mu\text{-fusion theorem}\ \} \\ & (S \cdot \mathsf{F}\ X \cdot R)^\circ = T \cdot \mathsf{F}\ (X^\circ) \cdot U \\ \equiv & \quad \{\ \text{converses; } \mathsf{F} \text{ is a relator (87)}\ \} \\ & R^\circ \cdot \mathsf{F}\ X^\circ \cdot S^\circ = T \cdot \mathsf{F}\ X^\circ \cdot U \\ \Leftarrow & \quad \{\ \text{Leibniz}\ \} \\ & R^\circ = T\ \wedge\ S^\circ = U \end{array}$$

So, we know how to compute the converse of a relational hylomorphism:

$$[\![S, R]\!]^\circ \quad = \quad [\![R^\circ, S^\circ]\!] \tag{121}$$

From (121) we easily infer that "*ana* is the converse of the *cata* of the converse":

$$\begin{array}{lcl} & & (\!|S|\!) \\ & = & [\![in, S]\!]^{\circ\circ} \\ & = & [\![S^\circ, in^\circ]\!]^\circ \\ & = & (\!|S^\circ|\!)^\circ \end{array}$$

Hylo(cata)-fusion:

$$V \cdot [\![S, R]\!] = [\![T, R]\!] \quad \Leftarrow \quad V \cdot S = T \cdot (\mathsf{F}\ V) \tag{122}$$

Proof: since $(V\cdot) = (V\backslash)^\flat$,

$$\begin{array}{ll} & V \cdot [\![S, R]\!] = [\![T, R]\!] \\ \Leftarrow & \quad \{\ \mu\text{-fusion theorem}\ \} \\ & V \cdot (S \cdot \mathsf{F}\ X \cdot R) = T \cdot \mathsf{F}\ (V \cdot X) \cdot R \\ \equiv & \quad \{\ \text{associative } (\cdot) \text{ and relator } \mathsf{F}\ \} \\ & (V \cdot S) \cdot \mathsf{F}\ X \cdot R = T \cdot (\mathsf{F}\ V) \cdot (\mathsf{F}\ X) \cdot R \\ \Leftarrow & \quad \{\ \text{Leibniz}\ \} \\ & V \cdot S = T \cdot (\mathsf{F}\ V) \end{array}$$

So, the following $\{\!|\_|\!\}$-fusion law

$$T \cdot \{\!|R|\!\} = \{\!|S|\!\} \quad \Leftarrow \quad T \cdot R = S \cdot (\mathsf{F}\, T)$$

arises from hylo(cata)-fusion (122).
Hylo(ana)-fusion:

$$[\!\![\, S, R \,]\!\!] \cdot V = [\!\![\, S, U \,]\!\!] \quad \Leftarrow \quad R \cdot V = \mathsf{F}\, V \cdot U \tag{123}$$

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

$$\begin{array}{ll} & [\!\![\, S, R \,]\!\!] \cdot V = [\!\![\, S, U \,]\!\!] \\ \Leftarrow & \quad \{\ \mu\text{-fusion theorem}\ \} \\ & (S \cdot \mathsf{F}\, X \cdot R) \cdot V = S \cdot \mathsf{F}\, (X \cdot V) \cdot U \\ \equiv & \quad \{\ \text{associative } (\cdot) \text{ and relator } \mathsf{F}\ \} \\ & S \cdot \mathsf{F}\, X \cdot (R \cdot V) = S \cdot (\mathsf{F}\, X) \cdot (\mathsf{F}\, V) \cdot U \\ \Leftarrow & \quad \{\ \text{Leibniz}\ \} \\ & R \cdot V = \mathsf{F}\, V \cdot U \end{array}$$

G Data Refinement — Getting Away With Recursion

How does one refine *recursive* VDM-SL models such as *eg.*

```
FS :: D: map Id to Node; -- FS means file system
Node = File | FS;        -- a Node is either a file
                         --   or a directory
Id = seq of char;        -- node identifiers
File :: F: seq of token  -- sequential files
```

that is, $FS = \mu\mathsf{F}$ for $\mathsf{F}\,X = Id \rightharpoonup (File + X)$:

$$\mu\mathsf{F} \underset{in}{\overset{out}{\cong}} Id \rightharpoonup (File + \mu\mathsf{F})$$

Two other examples of recursive data models follow: decision trees

```
DecTree :: question: What
           answers: map Answer to DecTree
What = seq of char;
Answer = seq of char;
```

— that is, $DecTree = \mu\mathsf{F}$ in

$$DecTree \quad \cong \quad What \times (Answer \rightharpoonup DecTree)$$

for $\mathsf{F}\,X = What \times (Answer \rightharpoonup X)$ — and formal expressions:

```
Exp    = Var | Term ;
Var    :: variable: Symbol ;
Term   :: operator: Symbol
          arguments: seq of Exp
          inv t == len t.arguments <= 20 ;
Symbol = seq of char
         inv s == len s <= 10 ;
```

— that is, $Exp = \mu\mathsf{F}$ in

$$Exp \quad \cong \quad Symbol + Symbol \times Exp^{\star}$$

for $\mathsf{F}\,X = Symbol + Symbol \times X^{\star}$.

Law for "getting away with recursion" Given

$$\mu\mathsf{F} \underset{in}{\overset{out}{\cong}} \mathsf{F}\,\mu\mathsf{F}$$

one has law (30):

$$\mu\mathsf{F} \underset{F}{\overset{\underbrace{(K \rightharpoonup \mathsf{F}\,K)}_{\text{“}heap''} \times K}{\leq}}$$

for K a data type of *"heap addresses"*, or *"pointers"*, such that $K \cong \mathbb{N}$.

Example of application Since

$$Exp = \mu X.(Symbol + Symbol \times X^{\star})$$

we have:

$$
\begin{array}{ll}
 & Exp \\
\leq & \quad \{ \text{ remove recursion } \} \\
 & (K \rightharpoonup (Symbol + Symbol \times K^{\star})) \times K \\
\leq & \quad \{ \text{ remove finite lists } \} \\
 & (K \rightharpoonup (Symbol + Symbol \times (\mathbb{N} \rightharpoonup K))) \times K \\
\leq & \quad \{ \text{ recall } A \rightharpoonup (B + C) \leq (A \rightharpoonup B) \times (A \rightharpoonup C) \ \} \\
 & (K \rightharpoonup Symbol) \times (K \rightharpoonup (Symbol \times (\mathbb{N} \rightharpoonup K))) \times K \\
\leq & \quad \{ \text{ remove nested } \rightharpoonup \ \} \\
 & (K \rightharpoonup Symbol) \times (K \rightharpoonup Symbol) \times ((\mathbb{N} \times K) \rightharpoonup K) \times K \\
\cong & \quad \{ \ A \times A \cong A^2 \ \} \\
 & (K \rightharpoonup Symbol)^2 \times ((\mathbb{N} \times K) \rightharpoonup K) \times K \\
\cong & \quad \{ \text{ recall } (C \rightharpoonup A)^B \cong B \times C \rightharpoonup A \} \\
 & \underbrace{((2 \times K) \rightharpoonup Symbol)}_{SYMBOLS} \times \underbrace{((\mathbb{N} \times K) \rightharpoonup K)}_{EXPRESSIONS} \times K
\end{array}
$$

In SQL, one will have a *symbols* table,

```
CREATE TABLE SYMBOLS (
       Symbol   CHAR    (20) NOT NULL,
       NodeId   NUMERIC (10) NOT NULL,
       IfVar    BOOLEAN      NOT NULL
       CONSTRAINT SYMBOLS_pk
                  PRIMARY KEY(NodeId,IfVar)
       );
```

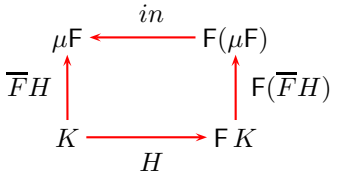
and an *expressions* table:

```
CREATE TABLE EXPRESSIONS (
       FatherId  NUMERIC (10) NOT NULL,
       ArgNr     NUMERIC (10) NOT NULL,
       ChildId   NUMERIC (10) NOT NULL
       CONSTRAINT EXPRESSIONS_pk
                  PRIMARY KEY (FatherId,ArgNr)
       );
```

However, can you *rely* on this implementation? We need details about *abstraction* invariant.

Abstraction function

- Main rôle in representation is played by simple F-*coalgebra* $K \rightharpoonup \mathsf{F}\ K$, understood as a (finite) piece of "linear storage", a "heap" or a "database" file.
- $\overline{F}$ — recall (62) — of type $(K \rightharpoonup \mu\mathsf{F})^{(K \rightharpoonup \mathsf{F}\ K)}$, is nothing but the F-anamorphism combinator:



$$
\begin{array}{lcl}
\overline{F} & = & [\![\_]\!]_{\mathsf{F}} \\
\overline{F}\ H & = & \mu X.\ in \cdot (\mathsf{F}\ X) \cdot H
\end{array}
$$

Partiality of implementation $F(H,k) = (\overline{F}H)k$ will be undefined wherever

- $k \notin \delta\, H$
- H is not "closed" over itself (see below)
- H is non-well-founded (see below)

Thus concrete invariant

$$\phi(H,k) \quad \stackrel{\text{def}}{=} \quad k \in \delta\, H \ \wedge\ (closed\ H) \ \wedge\ (wellf\ H)$$

In order to define $closed\ H$ and $wellf\ H$ we need H's *accessibility* relation $\prec_H$:

$$K \xleftarrow{\prec_H} K$$
$$\prec_H \quad \stackrel{\text{def}}{=} \quad \in_{\mathsf{F}} \cdot H$$

where $K \xleftarrow{\in_{\mathsf{F}}} \mathsf{F}\, K$ is the membership relation associated to F (97). Let $\prec_H^+$ denote the transitive closure of $\prec_H$. Then we define

$$closed\ H \quad = \quad \rho\ \prec_H^+ \subseteq \delta\, H$$

that is, all reacheable k are defined, and

$$wellf\ H \quad = \quad (\prec_H^+) \cap id = \bot$$

that is, $\prec_H^+$ is irreflexive (no cycles, no looping). For more details about this subject see section 10 of [5].

H Algorithmic Refinement — Getting Away With Recursion

A very common pattern in functional hylomorphisms is

$$\begin{array}{lcl} f & : & A \longrightarrow C \\ f & = & p \rightarrow b\ ,\ \theta \cdot \langle d, f \cdot e \rangle \end{array}$$

that is,

$$f \quad = \quad [b\ , \theta] \cdot (id + id \times f) \cdot (id + \langle d, e \rangle) \cdot p?$$

cf. diagram:

$$\begin{array}{ccccc} & & \overset{p?}{\longrightarrow} & \overset{id + \langle d, e \rangle}{\longrightarrow} & \\ A & & A + A & & A + B \times A \\ {\scriptstyle f}\downarrow & & & & \downarrow{\scriptstyle id + id \times f} \\ C & & \longleftarrow & & A + B \times C \\ & & \underset{[b\ , \theta]}{} & & \end{array}$$

By splitting $b = b_2 \cdot b_1$ and $d = d_2 \cdot d_1$ we obtain a more general intermediate type

$$\begin{array}{ccccc} & & \overset{p?}{\longrightarrow} & \overset{b_1 + \langle d_1, e \rangle}{\longrightarrow} & \\ A & & A + A & & D + B \times A \\ {\scriptstyle f}\downarrow & & & & \downarrow{\scriptstyle id + id \times f} \\ C & & \longleftarrow & & D + B \times C \\ & & \underset{[b_2\ , \theta]}{} & \underset{id + d_2 \times id}{} & \end{array}$$

In summary,

$$f \quad = \quad [\![\ [b_2\ , \theta \cdot (d_2 \times id)], (b_1 + \langle d_1, e \rangle) \cdot p?\ ]\!] \tag{124}$$

For instance, the membership test for finite lists

$$\begin{array}{ccccc} & & \overset{p?}{\longrightarrow} & \overset{! + \langle hd, e \rangle}{\longrightarrow} & \\ X^\star & & X^\star + X^\star & & 1 + X \times X^\star \\ {\scriptstyle f}\downarrow & & & & \downarrow{\scriptstyle id + id \times f} \\ 2 & & \longleftarrow & & 1 + X \times 2 \\ & & \underset{[\underline{\text{FALSE}}\ , \vee]}{} & \underset{id + (= x) \times id}{} & \end{array}$$

fits in this schema.

In many situations, we find that θ is associative, that is,

$$\theta \cdot \langle \theta \cdot \langle i, j \rangle, k \rangle = \theta \cdot \langle i, \theta \cdot \langle j, k \rangle \rangle$$

holds, for all i, j, k.

Exercise 17. Show that if θ is associative, then

$$\theta \cdot (f \times (\theta \cdot \langle h, k \rangle)) \quad = \quad \theta \cdot \langle \theta \cdot (f \times h), k \cdot \pi_2 \rangle \tag{125}$$

holds.

□

Associativity of θ leads us to a much simpler type scheme

$$\begin{array}{ccccc} & & \overset{p?}{\longrightarrow} & \overset{b_1 + \langle d, e \rangle}{\longrightarrow} & \\ A & & A + A & & D + C \times A \\ {\scriptstyle f}\downarrow & & & & \downarrow{\scriptstyle id + id \times f} \\ C & & \longleftarrow & & D + C \times C \\ & & \underset{[b_2\ , \theta]}{} & & \end{array}$$

Can we eliminate recursion from this hylomorphism, eg. convert it to a `while` loop?

What is a `while` loop? A `while` loop

$$\underline{while}\ p\ \underline{do}\ l \quad = \quad (\neg \cdot p) \to id\ ,\ (\underline{while}\ p\ \underline{do}\ l) \cdot l$$

is a very special case of hylomorphism $[\![\ R, S\ ]\!]$ where S does all the work and R does nothing

$$\begin{array}{ccc} & (id + l) \cdot (\neg \cdot p)? & \\ A & \longrightarrow & A + A \\ \underline{while}\ p\ \underline{do}\ l \downarrow & & \downarrow\ id + (\underline{while}\ p\ \underline{do}\ l) \\ A & \longleftarrow & A + A \\ & [id\ , id] & \end{array}$$

that is,

$$\underline{while}\ p\ \underline{do}\ l \quad = \quad \langle \mu\ f\ ::\ (\neg \cdot p) \to id\ ,\ f \cdot l \rangle \tag{126}$$

Our calculation plan is to convert the given hylo

$$f \quad = \quad p \to b\ ,\ \theta \cdot \langle d, f \cdot e \rangle$$

into

$$f \quad = \quad p \to b\ ,\ w \cdot \langle d, e \rangle$$

for some"tail recursive" w

$$w \quad = \quad x \to y\ ,\ w \cdot z$$

which will be nothing but the obvious `while`-loop:

$$w \quad = \quad y \cdot (\underline{while}\ (\neg \cdot x)\ \underline{do}\ z)$$

The unknowns of the problem are x, y and z.

Calculation Clearly,

$$\begin{array}{ll} & p \to b\ ,\ \theta \cdot \langle d, f \cdot e \rangle \\ = & \quad \{\ \times\text{-absorption}\ \} \\ & p \to b\ ,\ \underbrace{\theta \cdot (id \times f)}_{w} \cdot \langle d, e \rangle \end{array}$$

So, it suffices to find w such that

$$w \quad = \quad \theta \cdot (id \times f)$$

holds. For this we recall McCarthy-fusion laws

$$f \cdot (p \to g\ ,\ h) \quad = \quad p \to f \cdot g, f \cdot h \tag{127}$$

$$(p \to f, g) \cdot h \quad = \quad (p \cdot h) \to (f \cdot h), (g \cdot h) \tag{128}$$

$$\langle f, (p \to g\ ,\ h) \rangle \quad = \quad p \to \langle f, g \rangle\ ,\ \langle h, f \rangle \tag{129}$$

$$f \times (p \to g\ ,\ h) \quad = \quad p \cdot \pi_2 \to f \times g\ ,\ f \times h \tag{130}$$

and calculate:

$$\begin{array}{ll} & \langle \mu\ f\ ::\ \underbrace{p \to b\ ,\ \theta \cdot \langle d, f \cdot e \rangle}_{\mathsf{F}\,f} \rangle \\ = & \quad \{\ \text{square rule, } \langle \mu\ f\ ::\ \mathsf{F}\ f \rangle = \langle \mu\ f\ ::\ \mathsf{F}^2\ f \rangle\ (115)\ \} \\ & \langle \mu\ f\ ::\ p \to b\ ,\ \theta \cdot \langle d, (p \to b\ ,\ \theta \cdot \langle d, f \cdot e \rangle) \cdot e \rangle \rangle \\ = & \quad \{\ \text{(reverse) } \times\text{-absorption}\ \} \end{array}$$

$$
\begin{array}{ll}
& \langle \mu\, f \;::\; p \to b\;,\; \theta \cdot (id \times (p \to b\;,\; \theta \cdot \langle d, f \cdot e \rangle) \cdot \langle d, e \rangle \rangle \\
= & \quad \{ \text{ McCarthy-fusion law (130) } \} \\
& \langle \mu\, f \;::\; p \to b\;,\; \theta \cdot (p \cdot \pi_2 \to id \times b\;,\; id \times (\theta \cdot \langle d, f \cdot e \rangle)) \cdot \langle d, e \rangle \rangle \\
= & \quad \{ \text{ McCarthy-fusion law (127) } \} \\
& \langle \mu\, f \;::\; \begin{array}{l} p \to b\;, \\ (p \cdot \pi_2 \to \theta \cdot (id \times b)\;,\; \theta \cdot (id \times (\theta \cdot \langle d, f \cdot e \rangle))) \cdot \langle d, e \rangle \end{array} \rangle \\
= & \quad \{ \theta \text{ is associative (125) } \} \\
& \langle \mu\, f \;::\; \begin{array}{l} p \to b\;, \\ \underbrace{p \cdot \pi_2 \to \theta \cdot (id \times b)\;,\; \theta \cdot \langle \theta \cdot (id \times d), f \cdot e \cdot \pi_2 \rangle}_{\text{candidate } w} \cdot \langle d, e \rangle \end{array} \rangle
\end{array}
$$

We now focus on the candidate w:

$$
\begin{array}{rcl}
w & = & p \cdot \pi_2 \to \theta \cdot (id \times b)\;,\; \theta \cdot \langle \theta \cdot (id \times d), f \cdot e \cdot \pi_2 \rangle \\
& = & \quad \{ \times\text{-absorption } \} \\
& & p \cdot \pi_2 \to \theta \cdot (id \times b)\;,\; \theta \cdot (id \times f) \cdot \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle \\
& = & \quad \{ \text{ pattern matching (recall assumption } w = \theta \cdot (id \times f)) \; \} \\
& & \underbrace{\underbrace{p \cdot \pi_2}_{x} \to \underbrace{\theta \cdot (id \times b)}_{y}\;,\; \underbrace{\theta \cdot (id \times f)}_{w} \cdot \underbrace{\langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle}_{z}}_{w}
\end{array}
$$

So we've found tail-recursive solution

$$
w \;=\; p \cdot \pi_2 \to \theta \cdot (id \times b)\;,\; w \cdot \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle
$$

Altogether:

$$
\begin{array}{rcl}
f & = & p \to b\;,\; w \cdot \langle d, e \rangle \\
w & = & p \cdot \pi_2 \to \theta \cdot (id \times b)\;,\; w \cdot \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle
\end{array}
$$

Quite often $b = \underline{u}$, where u is the unit of θ (that is, $a\;\theta\;u = a$ for all a). In this case,

$$
\theta \cdot (f \times b) = f \cdot \pi_1 \tag{131}
$$

This makes w simpler:

$$
\begin{array}{rcl}
w & = & p \cdot \pi_2 \to \pi_1\;,\; w \cdot \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle \\
& = & \pi_1 \cdot (\underline{while}\; \neg \cdot p \cdot \pi_2\; \underline{do}\; \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle)
\end{array}
$$

Example: factorial From

$$
fac \;=\; (=_0) \to \underline{1}\;,\; mul \cdot \langle id, fac \cdot pred \rangle
$$

we get VDM-SL functions

```
fac(n) == if n=0 then 1 else w(n,n-1)
w(m,i) == if i=0 then m else w(m * i, i-1)
```

Below we show how the `while` loop can be made explicit by moving from funcions to operations:

```
fac : nat ==> nat
fac(n) ==
    if n=0 then return 1
```

```
    else (dcl m: nat := n,
              i: nat := n -1;
          while (i <> 0) do
                (m := m * i; i := i-1);
          return m;
);
```

Other results For $b = \underline{u}$ there is another recusion removal law which we could prove similarly:

$$\begin{array}{lcl} f & : & A \longrightarrow C \\ f & = & p \rightarrow \underline{u}\ ,\ \theta \cdot \langle d, f \cdot e \rangle \end{array}$$

is equivalent to

$$\begin{array}{lcl} f & = & w \cdot \langle \underline{u}, id \rangle \\ w & = & p \cdot \pi_2 \rightarrow \pi_1\ ,\ w \cdot \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle \end{array}$$

The corresponding VDM-SL version of factorial is as follows:

```
fac : nat ==> nat
fac(n) ==
    (dcl m: nat := 1,
         i: nat := n;
     while (i <> 0) do
           (m := m * i; i := i - 1);
     return m
);
```

`For` loops VDM-SL/VDM++ syntax includes for loops. Therefore, the two while loops above can be converted into, respectively,

```
fac(n) ==
    if n=0 then return 1
    else (dcl m: nat := n;
          for i=n-1 to 1 by -1 do m := m * i;
         return m;
);
```

and

```
fac(n) ==
    (dcl m: nat := 1;
     for i = n to 1 by -1 do m := m * i;
     return m
);
```

In summary, we have the following results for iterative factorization of linear recursion:

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

$$\begin{array}{lcll} \langle \mu\, f\ ::\ p \rightarrow b\ ,\ \theta \cdot \langle d, f \cdot e \rangle \rangle & = & p \rightarrow b\ ,\ \theta \cdot (id \times b) \cdot w \cdot \langle d, e \rangle & (132) \\ & & where & \\ & & w = \underline{while}\ (\neg \cdot p \cdot \pi_2)\ \underline{do}\ \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle & \end{array}$$

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

$$\langle \mu\, f\ ::\ p \rightarrow \underline{u}\ ,\ \theta \cdot \langle d, f \cdot e \rangle \rangle \quad = \quad \pi_1 \cdot w \cdot \langle \underline{u}, id \rangle \qquad (133)$$

where w is the same as in (132) above.

I Solutions to some of the exercises

Resolução 4: Let N and P_0 denote the corelexives $\lceil \lambda i.i < 0 \rceil$ and $\lceil \lambda i.i \geq 0 \rceil$, respectively. Then the PF-transform of Abs is

$$P_0 \cdot (id \cup sym) \tag{134}$$

and that of abs is $N \rightarrow sym\ ,\ id$. Clearly,

$$\begin{aligned} sym &= sym^\circ && (135)\\ N \cdot P_0 &= \bot && (136)\\ N \cdot N_0 &= N && (137)\\ N_0 \cdot sym &= sym \cdot P_0 && (138)\\ P_0 \cdot N_0 &= \underline{0} \cdot \underline{0}^\circ && (139) \end{aligned}$$

where N_0 has the obvious meaning. Then complete the steps left unjustified in:

$$\begin{array}{ll} & Abs \vdash abs \\ \Leftarrow & \quad \{\ (18)\ \} \\ & abs \cdot Abs^\circ \subseteq id \\ \equiv & \quad \{\ \text{in-lining PF-transforms ; converses ; } sym \text{ is symmetric}\ \} \\ & (N \rightarrow sym\ ,\ id) \cdot (id \cup sym) \cdot P_0 \subseteq id \\ \equiv & \quad \{\ \text{conditional (7) ; negation of } N \text{ is } P_0 \text{ ; distribution and (138)}\ \} \\ & (sym \cdot N \cup P_0) \cdot (P_0 \cup N_0 \cdot sym) \subseteq id \\ \equiv & \quad \{\ \ldots\ \} \\ & sym \cdot N \cdot sym \cup P_0 \cup \underline{0} \cdot \underline{0}^\circ \cdot sym \subseteq id \\ \equiv & \quad \{\ \ldots\ \} \\ & sym \cdot N \subseteq sym \wedge \underline{0}^\circ \subseteq \underline{0}^\circ \cdot sym \\ \equiv & \quad \{\ \ldots\ \} \\ & \underline{0} = sym \cdot \underline{0} \\ \equiv & \quad \{\ \text{going pointwise}\ \} \\ & 0 = -0 \end{array}$$

□

J PF-transform Reference Manual

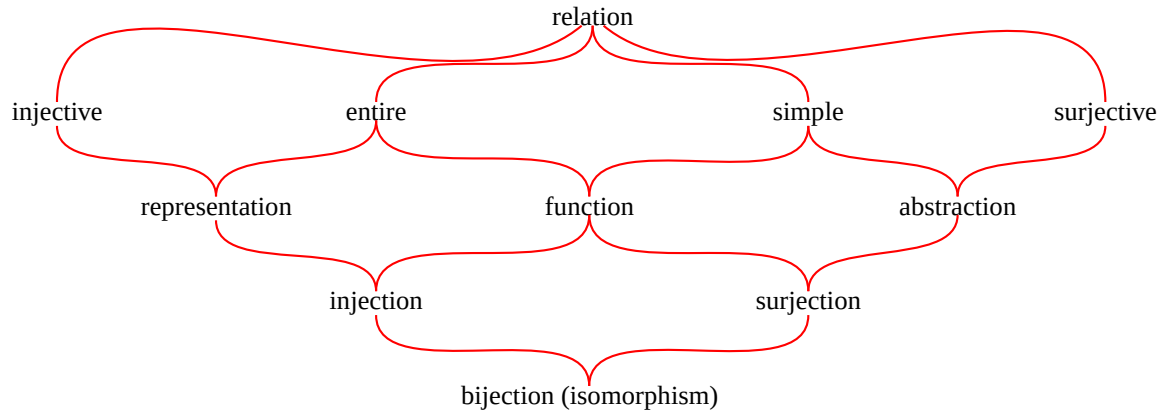
J.1 Relational taxonomy

Classification criteria:

	Reflexive	*Coreflexive*
ker R	entire R	injective R
img R	surjective R	simple R

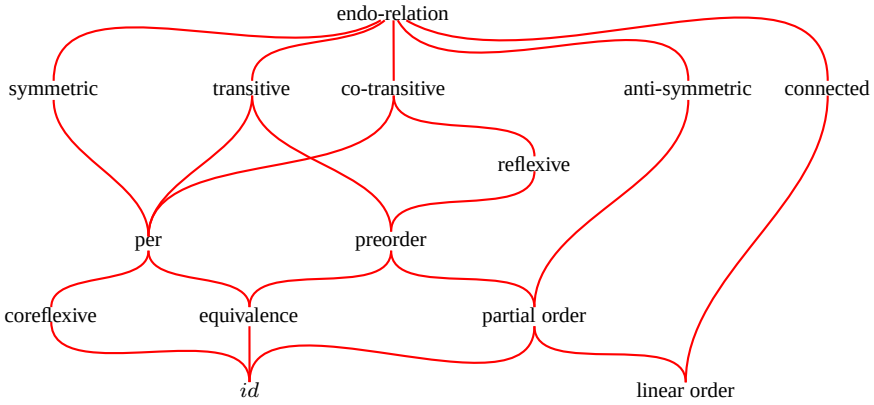
(140)

Binary relations:



(141)

Orders:



(142)

J.2 PF-transformation rules

"Guardanapo":

$$b(f^\circ \cdot R \cdot g)a \quad \equiv \quad (f\ b)R(g\ a) \tag{143}$$

Left-division:

$$b\ (R \setminus Y)\ a \quad \equiv \quad \langle \forall\ c\ :\ c\ R\ b :\ c\ Y\ a \rangle \tag{144}$$

Pointwise ordering on functions:

$$f\ \dot{\sqsubseteq}\ g \quad \equiv \quad f \subseteq \sqsubseteq \cdot g \quad \equiv \quad \langle \forall\ a\ ::\ (f\ a) \sqsubseteq (g\ a) \rangle \tag{145}$$

J.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	$(h\cdot)$	$(h^\circ\cdot)$	NB: h is a function
"converse" shunting rule	$(\cdot h^\circ)$	$(\cdot h)$	NB: h is a function
left-division	$(R\cdot)$	$(R \setminus\ )$	R under ...
right-division	$(\cdot R)$	$(\ /R)$	... over R
range	ρ	$(\cdot \top)$	lower $\subseteq$ restricted to coreflexives
domain	δ	$(\top \cdot)$	lower $\subseteq$ restricted to coreflexives
implication	$(R \cap\ )$	$(R \Rightarrow\ )$	Note that $(R \Rightarrow) = (\neg R \cup\ )$
difference	$(\_ - R)$	$(R \cup\ )$	
PROPERTIES			
cancellation	$X \subseteq (g \cdot f)X$	$(f \cdot g)Y \subseteq Y$	
definition	$f\ X = \bigcap\{Y \mid X \subseteq gY\}$	$g\ Y = \bigcup\{X \mid f\ X \subseteq Y\}$	
distribution	$f(X \cup Y) = (f\ X) \cup (f\ Y)$	$g(X \cap Y) = (g\ X) \cap (g\ Y)$	$f(\bigcup_i X_i) = \bigcup_i (f\ X_i)$ $g(\bigcap_i X_i) = \bigcap_i (g\ X_i)$

(146)

J.4 Other Galois connections

Meet-universal

$$X \subseteq (R \cap S) \quad \equiv \quad (X \subseteq R) \wedge (X \subseteq S) \tag{147}$$

Join-universal

$$(R \cup S) \subseteq X \quad \equiv \quad (R \subseteq X) \wedge (S \subseteq X) \tag{148}$$

Split-universal

$$X \subseteq \langle R, S \rangle \quad \equiv \quad \pi_1 \cdot X \subseteq R \ \wedge \ \pi_2 \cdot X \subseteq S \tag{149}$$

Either-universal

$$X = [R\,, S] \quad \equiv \quad X \cdot i_1 = R \quad \wedge \quad X \cdot i_2 = S \tag{150}$$

J.5 "Almost" Galois connections

"Shunting" rules for S a simple relation:

$$S \cdot R \subseteq T \quad \equiv \quad (\delta\, S) \cdot R \quad \subseteq \quad S^\circ \cdot T \tag{151}$$
$$R \cdot S^\circ \subseteq T \quad \equiv \quad R \cdot \delta\, S \quad \subseteq \quad T \cdot S \tag{152}$$

Variants concerning domain and range:

$$\delta\, R \subseteq X \quad \equiv \quad R \subseteq R \cdot X \tag{153}$$
$$\rho\, R \subseteq X \quad \equiv \quad R \subseteq X \cdot R \tag{154}$$

J.6 Converses

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \tag{155}$$

J.7 Coreflexives

Since coreflexives are simple, the following follow from (151,152):

$$\Phi \cdot R \subseteq S \quad \equiv \quad \Phi \cdot R \subseteq \Phi \cdot S \tag{156}$$
$$R \cdot \Phi \subseteq S \quad \equiv \quad R \cdot \Phi \subseteq S \cdot \Phi \tag{157}$$

Domain and range:

$$\delta\,(R \cap S) \quad = \quad (R^\circ \cdot S) \cap id \tag{158}$$
$$\delta\,(R \cdot S) \quad = \quad \delta\,(\delta\, R \cdot S) \tag{159}$$
$$\rho\, R \quad = \quad \delta\,(R^\circ) \tag{160}$$

Therefore, for Φ coreflexive

$$\delta\, \Phi = \Phi \tag{161}$$

Also a consequence of the above:

$$\delta\, R \quad = \quad \mathsf{ker}\, R \cap id \tag{162}$$
$$\rho\, R \quad = \quad \mathsf{img}\, R \cap id \tag{163}$$

J.8 Relational divisions

$$(R \setminus S) \cdot f = R \setminus (S \cdot f) \tag{164}$$

J.9 Meets

$$(S \cap T) \cdot R = (S \cdot R) \cap (T \cdot R) \quad \Leftarrow \quad T \cdot \mathsf{img}\, R \subseteq T \vee S \cdot \mathsf{img}\, R \subseteq S \tag{165}$$

Therefore, for f a function,

$$(S \cap T) \cdot f = (S \cdot f) \cap (T \cdot f) \tag{166}$$

$$R \cdot (S \cap T) = (R \cdot S) \cap (R \cdot T) \quad \Leftarrow \quad (\mathsf{ker}\, R) \cdot T \subseteq T \vee (\mathsf{ker}\, R) \cdot S \subseteq S \tag{167}$$

J.10 Splits

Definition equivalent to (149)

$$\langle R, S\rangle \quad = \quad \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{168}$$

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

$$(a, b)\langle R, S\rangle c \quad \equiv \quad a\ R\ c\ \wedge\ b\ S\ c \tag{169}$$

Split cancellation

$$\pi_1 \cdot \langle R, S\rangle = R \cdot \delta\, S \quad \wedge \quad \pi_2 \cdot \langle R, S\rangle = S \cdot \delta\, R \tag{170}$$

Split (conditional) fusion [1]:

$$\langle R, S\rangle \cdot T = \langle R \cdot T, S \cdot T\rangle \quad \Leftarrow \quad R \cdot (\mathsf{img}\, T) \subseteq R \vee S \cdot (\mathsf{img}\, T) \subseteq S \tag{171}$$

Split absorption

$$\langle R \cdot T, S \cdot U\rangle \quad = \quad (R \times S) \cdot \langle T, U\rangle \tag{172}$$

Splits and converses:

$$\langle R, S\rangle^\circ \cdot \langle X, Y\rangle \quad = \quad (R^\circ \cdot X) \cap (S^\circ \cdot Y) \tag{173}$$

Therefore:

$$\mathsf{ker}\, \langle R, S\rangle \quad = \quad \mathsf{ker}\, R \cap \mathsf{ker}\, S \tag{174}$$

J.11 Eithers

Definition:

$$[R\ , S] \quad = \quad (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \tag{175}$$

From (150), all coproduct properties extend to relations, in particular: $+$-reflexion:

$$id = [i_1\ , i_2] \tag{176}$$

etc. Eithers and converses:

$$[R\ , S] \cdot [T\ , U]^\circ \quad = \quad (R \cdot T^\circ) \cup (S \cdot U^\circ) \tag{177}$$

J.12 Relational projection

Definition

$$\pi_{g,f} R \quad \stackrel{\text{def}}{=} \quad g \cdot R \cdot f^\circ \tag{178}$$

Property

$$\pi_{g,f} R \subseteq S \quad \equiv \quad g(S \leftarrow R)f \tag{179}$$

Wherever M is a simple, finite relation (coflexives included), and $projg, fM$ is also simple, this can be written pointwise as follows

$$\pi_{g,f} M \quad = \quad \{f\ a \mapsto g(M\ a)\ |\ a \in\ dom\ M\} \tag{180}$$

[1] Theorem 12.30 in [1].

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] 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.

[3] Paul Hoogendijk. *A Generic Theory of Data Types*. PhD thesis, University of Eindhoven, The Netherlands, 1997.

[4] C.B. Jones. *Systematic Software Development Using VDM*. Series in Computer Science. Prentice-Hall International, 1990. 1st edition (1986), 2nd edition (1990). PDF available from `http://www.vdmportal.org/twiki/bin/view/Main/Jonesbook`.

[5] J.N. Oliveira. Data transformation by calculation, June 2007. Tutorial notes for GTTSE'07.

[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.