Explicit Substitutions à la de Bruijn: the local and global way

Fairouz Kamareddine
Joint work with
Alejandro Ríos
`http://www.macs.hw.ac.uk/~fairouz/talks/talks2003/mlcestalk03.ps`

5 July 2003

The $\lambda\sigma$-calculus

Terms	$\Lambda\sigma^t ::= \mathtt{1} \mid \Lambda\sigma^t\Lambda\sigma^t \mid \lambda\Lambda\sigma^t \mid \Lambda\sigma^t[\Lambda\sigma^s]$
Substitutions	$\Lambda\sigma^s ::= id \mid \uparrow \mid \Lambda\sigma^t \cdot \Lambda\sigma^s \mid \Lambda\sigma^s \circ \Lambda\sigma^s$

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
(VarId)	$\mathtt{1}\,[id]$	$\longrightarrow$	$\mathtt{1}$
(VarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\mathtt{1} \cdot (s \circ \uparrow)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(IdL)	$id \circ s$	$\longrightarrow$	s
(ShiftId)	$\uparrow \circ\, id$	$\longrightarrow$	$\uparrow$
(ShiftCons)	$\uparrow \circ\, (a \cdot s)$	$\longrightarrow$	s
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(Ass)	$(s_1 \circ s_2) \circ s_3$	$\longrightarrow$	$s_1 \circ (s_2 \circ s_3)$

We can code $\mathtt{n}$ by the term $\mathtt{1}[\uparrow^{n-1}]$.

The $\lambda\upsilon$-rules

$$\Lambda\upsilon^t ::= I\!N \mid \Lambda\upsilon^t\Lambda\upsilon^t \mid \lambda\Lambda\upsilon^t \mid \Lambda\upsilon^t[\Lambda\upsilon^s]$$
$$\Lambda\upsilon^s ::= \uparrow \mid \Uparrow(\Lambda\upsilon^s) \mid \Lambda\upsilon^t.$$

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b/]$
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\Uparrow(s)])$
(FVar)	$\mathtt{1}\,[a/]$	$\longrightarrow$	a
(RVar)	$\mathtt{n}+\mathtt{1}\,[a/]$	$\longrightarrow$	$\mathtt{n}$
(FVarLift)	$\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(RVarLift)	$\mathtt{n}+\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{n}\,[s]\,[\uparrow]$
(VarShift)	$\mathtt{n}\,[\uparrow]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}$

The $\lambda\sigma_{\Uparrow}$-rules

$$\Lambda\sigma^t_{\Uparrow} ::= I\!N \mid \Lambda\sigma^t_{\Uparrow}\Lambda\sigma^t_{\Uparrow} \mid \lambda\Lambda\sigma^t_{\Uparrow} \mid \Lambda\sigma^t_{\Uparrow}[\Lambda\sigma^s_{\Uparrow}]$$

$$\Lambda\sigma^s_{\Uparrow} ::= id \mid \uparrow \mid \Uparrow(\Lambda\sigma^s_{\Uparrow}) \mid \Lambda\sigma^t_{\Uparrow} \cdot \Lambda\sigma^s_{\Uparrow} \mid \Lambda\sigma^s_{\Uparrow} \circ \Lambda\sigma^s_{\Uparrow}.$$

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\Uparrow(s)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(Varshift1)	$\mathtt{n}\,[\uparrow]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}$
(Varshift2)	$\mathtt{n}\,[\uparrow \circ s]$	$\longrightarrow$	$\mathtt{n}+\mathtt{1}\,[s]$
(FVarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(RVarCons)	$\mathtt{n}+\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	$\mathtt{n}\,[s]$
(FVarLift1)	$\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(FVarLift2)	$\mathtt{1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{1}\,[t]$

(RVarLift1)	$\mathtt{n+1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{n}[s \circ \uparrow]$
(RVarLift2)	$\mathtt{n+1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{n}[s \circ (\uparrow \circ t)]$
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(ShiftLift1)	$\uparrow \circ \Uparrow(s)$	$\longrightarrow$	$s \circ \uparrow$
(ShiftLift2)	$\uparrow \circ (\Uparrow(s) \circ t)$	$\longrightarrow$	$s \circ (\uparrow \circ t)$
(Lift1)	$\Uparrow(s) \circ \Uparrow(t)$	$\longrightarrow$	$\Uparrow(s \circ t)$
(Lift2)	$\Uparrow(s) \circ (\Uparrow(t) \circ u)$	$\longrightarrow$	$\Uparrow(s \circ t) \circ u$
(LiftEnv)	$\Uparrow(s) \circ (a \cdot t)$	$\longrightarrow$	$a \cdot (s \circ t)$
(IdL)	$id \circ s$	$\longrightarrow$	s
(IdR)	$s \circ id$	$\longrightarrow$	s
(LiftId)	$\Uparrow(id)$	$\longrightarrow$	id
(Id)	$a\,[id]$	$\longrightarrow$	a

Lambda calculus with de Bruijn indices

- $\Lambda ::= \mathbb{N} \mid (\Lambda\Lambda) \mid (\lambda\Lambda)$ $\qquad (\lambda A)\, B \to_\beta A\{\!\{\mathtt{1} \leftarrow B\}\!\}$

- *meta-updatings* $U_k^i : \Lambda \to \Lambda$ for $k \geq 0$ and $i \geq 1$:

  $U_k^i(AB) \equiv U_k^i(A)\, U_k^i(B) \qquad U_k^i(\lambda A) \equiv \lambda(U_{k+1}^i(A))$

  $$U_k^i(\mathtt{n}) \equiv \begin{cases} \mathtt{n}+\mathtt{i}-\mathtt{1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \leq k\,. \end{cases}$$

- *meta-substitutions* at level $i \geq 1$, of a term $B \in \Lambda$ in a term $A \in \Lambda$:

$$
\begin{aligned}
(A_1 A_2)\{\!\{\mathtt{i} \leftarrow B\}\!\} &\equiv (A_1\{\!\{\mathtt{i} \leftarrow B\}\!\})\,(A_2\{\!\{\mathtt{i} \leftarrow B\}\!\}) \\
(\lambda A)\{\!\{\mathtt{i} \leftarrow B\}\!\} &\equiv \lambda(A\{\!\{\mathtt{i}+\mathtt{1} \leftarrow B\}\!\}) \\
\mathtt{n}\{\!\{\mathtt{i} \leftarrow B\}\!\} &\equiv \begin{cases} \mathtt{n}-\mathtt{1} & \textsf{if}\ \ n > i \\ U_0^i(B) & \textsf{if}\ \ n = i \\ \mathtt{n} & \textsf{if}\ \ n < i\,. \end{cases}
\end{aligned}
$$

- **Lemma 1.**

  – $$\begin{array}{lcll} U_k^i(A)\{\!\{\mathtt{n}\leftarrow B\}\!\} & \equiv & U_k^{i-1}(A) & \textit{if } k < n < k+i \\ U_k^i(A)\{\!\{\mathtt{n}\leftarrow B\}\!\} & \equiv & U_k^i(A\{\!\{\mathtt{n}-\mathtt{i}+\mathtt{1}\leftarrow B\}\!\}) & \textit{if } k+i < n \end{array}$$

  – $$\begin{array}{lcll} U_k^i(U_p^j(A)) & \equiv & U_p^{j+i-1}(A) & \textit{if } p \leq k < j+p \\ U_k^i(U_p^j(A)) & \equiv & U_p^j(U_{k+1-j}^i(A)) & \textit{if } j+p \leq k+1 \end{array}$$

  – Meta-substitution lemma *For* $1 \leq i \leq n$ *we have:*
  $A\{\!\{\mathtt{i}\leftarrow B\}\!\}\{\!\{\mathtt{n}\leftarrow C\}\!\} \equiv A\{\!\{\mathtt{n}+\mathtt{1}\leftarrow C\}\!\}\{\!\{\mathtt{i}\leftarrow B\{\!\{\mathtt{n}-\mathtt{i}+\mathtt{1}\leftarrow C\}\!\}\}\!\}$.

  – Distribution lemma
  For $n \leq k+1$ *we have:* $U_k^i(A\{\!\{\mathtt{n}\leftarrow B\}\!\}) \equiv U_{k+1}^i(A)\{\!\{\mathtt{n}\leftarrow U_{k-n+1}^i(B)\}\!\}$.

The λs-calculus

$$\Lambda s ::= I\!N \mid \Lambda s \Lambda s \mid \lambda \Lambda s \mid \Lambda s\, \sigma^j \Lambda s \mid \varphi^i_k \Lambda s \qquad \textit{where} \quad j,\, i \geq 1\,, \quad k \geq 0\,.$$

σ-generation	$(\lambda a)\, b$	$\longrightarrow$	$a\, \sigma^1\, b$
σ-λ-transition	$(\lambda a)\, \sigma^j b$	$\longrightarrow$	$\lambda(a \sigma^{j+1} b)$
σ-app-transition	$(a_1\, a_2)\, \sigma^j b$	$\longrightarrow$	$(a_1\, \sigma^j b)\, (a_2\, \sigma^j b)$
σ-destruction	$\mathtt{n}\, \sigma^j b$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \text{if } n > j \\ \varphi_0^j\, b & \text{if } n = j \\ \mathtt{n} & \text{if } n < j \end{cases}$
φ-λ-transition	$\varphi_k^i(\lambda a)$	$\longrightarrow$	$\lambda(\varphi_{k+1}^i\, a)$
φ-app-transition	$\varphi_k^i(a_1\, a_2)$	$\longrightarrow$	$(\varphi_k^i\, a_1)\, (\varphi_k^i\, a_2)$
φ-destruction	$\varphi_k^i\, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \leq k \end{cases}$

The extra rules of the λs_e-calculus

- $\Lambda s_{op} ::= \mathbf{V} \mid I\!N \mid \Lambda s_{op}\Lambda s_{op} \mid \lambda\Lambda s_{op} \mid \Lambda s_{op}\,\sigma^j\Lambda s_{op} \mid \varphi^i_k\Lambda s_{op}$

- *Loss of confluence*

$$(X\sigma^1 Y)\sigma^1 1 \leftarrow ((\lambda X)Y)\sigma^1 1 \rightarrow ((\lambda X)\sigma^1 1)(Y\sigma^1 1)$$

$(X\sigma^1 Y)\sigma^1 1$ and $((\lambda X)\sigma^1 1)(Y\sigma^1 1)$ *have no common reduct*

σ-σ-*transition*	$(a\,\sigma^i b)\,\sigma^j\,c$	$\longrightarrow$	$(a\,\sigma^{j+1}\,c)\ \sigma^i\,(b\,\sigma^{j-i+1}\,c)$	if	$i \leq j$
σ-φ-*transition 1*	$(\varphi^i_k\,a)\,\sigma^j\,b$	$\longrightarrow$	$\varphi^{i-1}_k\,a$	if	$k < j < k+i$
σ-φ-*transition 2*	$(\varphi^i_k\,a)\,\sigma^j\,b$	$\longrightarrow$	$\varphi^i_k(a\,\sigma^{j-i+1}\,b)$	if	$k+i \leq j$
φ-σ-*transition*	$\varphi^i_k(a\,\sigma^j\,b)$	$\longrightarrow$	$(\varphi^i_{k+1}\,a)\,\sigma^j\,(\varphi^i_{k+1-j}\,b)$	if	$j \leq k+1$
φ-φ-*transition 1*	$\varphi^i_k\,(\varphi^j_l\,a)$	$\longrightarrow$	$\varphi^j_l\,(\varphi^i_{k+1-j}\,a)$	if	$l+j \leq k$
φ-φ-*transition 2*	$\varphi^i_k\,(\varphi^j_l\,a)$	$\longrightarrow$	$\varphi^{j+i-1}_l\,a$	if	$l \leq k < l+j$

- For every $\xi \in \{\sigma, \sigma_{\Uparrow}, \upsilon, s\}$, ξ is SN and $\lambda\xi$ is confluent on closed terms.

- Only $\lambda\sigma_{\Uparrow}$ and the λs_e are confluent on open terms

- Only $\lambda\upsilon$ and λs have Preservation of Strong Normalisation (PSN)

- λs has an extension λs_e which is confluent on open terms, but $\lambda\upsilon$ does not.

- Is s_e Strongly Normalising? We know s_e Weakly Normalising.

- We have fully proof checked the proof of SN of σ in ALF, we have investigated different termination techniques, but are still unable to show SN of s_e.

Item Notation/Lambda Calculus à la de Bruijn

- $\mathcal{I}$ translates to item notation:

$$\mathcal{I}(x) = x, \qquad \mathcal{I}(\lambda x.B) = [x]\mathcal{I}(B), \qquad \mathcal{I}(AB) = \langle\mathcal{I}(B)\rangle\mathcal{I}(A)$$

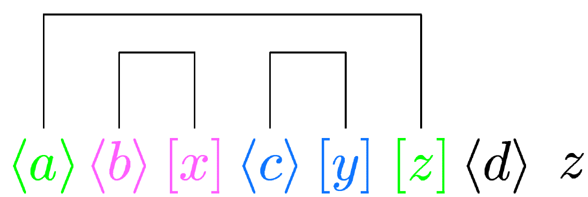
- $(\lambda x.\lambda y.xy)z$ translates to $\langle z\rangle[x][y]\langle y\rangle x$.
- The *wagons* are $\langle z\rangle$, $[x]$, $[y]$ and $\langle y\rangle$. The last x is the *heart* of the term.
- The *applicator wagon* $\langle z\rangle$ and *abstractor wagon* $[x]$ occur NEXT to each other.

- The β rule $(\lambda x.A)B \to_\beta A[x := B]$ becomes in item notation:

$$\langle B\rangle[x]A \to_\beta [x := B]A$$

Redexes in Item Notation

Classical Notation	Item Notation
$\underline{((\lambda_x.(\lambda_y.\lambda_z.zd)c)b)}a$	$\langle a\rangle \underline{\langle b\rangle [x]}\langle c\rangle [y][z]\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{((\lambda_y.\lambda_z.zd)c)}a$	$(a)\underline{\langle c\rangle [y]}[z]\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{(\lambda_z.zd)a}$	$\underline{\langle a\rangle [z]}(d)z$
$\downarrow_\beta$	$\downarrow_\beta$
ad	$\langle d\rangle a$



Automath

- Mathematical text in AUTOMATH written as a finite list of *lines* of the form:

  $x_1 : A_1, \ldots, x_n : A_n \vdash g(x_1, \ldots, x_n) = t : T.$

  Here g is a new name, an abbreviation for the expression t of type T and $x_1, \ldots, x_n$ are the parameters of g, with respective types $A_1, \ldots, A_n$.

- Each line introduces a new definition which is inherently parametrised by the variables occurring in the context needed for it.

- If line $x_1 : A_1, \ldots, x_n : A_n \vdash g(x_1, \ldots, x_n) = t : T$ occurs in a book $\mathfrak{B}$ then we can unfold the definition by: $b(\Sigma_1, \ldots, \Sigma_n) \to_\delta \Xi_1[x_1, \ldots, x_n{:=}\Sigma_1, \ldots, \Sigma_n]$.

- Developments of ordinary mathematical theory in AUTOMATH (van Benthem Jutting) revealed that this combined definition and parameter mechanism is vital for keeping proofs manageable and sufficiently readable for humans.

$\Delta\Lambda$

- In AUT-SL, de Bruijn described how a complete AUTOMATH book can be written as a single λ-calculus formula.
- *Disadvantage of AUT-SL:* in order to put the book into the λ-calculus framework, we must first eliminate all definitional lines of the book.
- De Bruijn did not like this: without definitions, formulae grow exponentially.
- For this reason, de Bruijn developed the $\Delta\Lambda$ with which he wanted to embrace all essential aspects of AUTOMATH apart from type inclusion.
- $\Delta\Lambda$ is the lambda calculus written in his wagon notation (as above).

- In $\Delta\Lambda$, de Bruijn favours trees over character strings and does not make use of AT-couples.

Local versus Global reductions

- In $\Delta\Lambda$, de Bruijn replaced β-reduction by a sequence of local β-reductions and AT-removals.

- The reason for this is that the delta reductions $\rightarrow_\delta$ of AUTOMATH can be considered as local β-reductions, and not as ordinary β-reductions.

- De Bruijn defined local β-reduction, which keeps the AT-pair and does β-reduction at one instance (instead of all the instances).

- Example

$$\langle y\rangle[x]\langle y\rangle x \leftarrow_{L\beta} \langle y\rangle[x]\langle x\rangle x \rightarrow_{L\beta} \langle y\rangle[x]\langle x\rangle y$$

- Doing a further local β-reduction gives
$\langle y\rangle[x]\langle y\rangle y \leftarrow_{L\beta} \langle y\rangle[x]\langle y\rangle x \leftarrow_{L\beta} \langle y\rangle[x]\langle x\rangle x \rightarrow_{L\beta} \langle y\rangle[x]\langle x\rangle y \rightarrow_{L\beta} \langle y\rangle[x]\langle y\rangle y$

- Now we can remove the AT-pair $\langle y\rangle[x]$ from $\langle y\rangle[x]\langle y\rangle y$ obtaining $\langle y\rangle y$.

A calculus of local explicit substitutions

- In order to treat local substitution, Kamareddine and Nederpelt proposed:

$\sigma_{0\delta}$*-transition*	$(c\,\sigma^i)(b\,\delta)a$	$\longrightarrow$	$((c\,\sigma^i)b\,\delta)a$	
$\sigma_{1\delta}$*-transition*	$(c\,\sigma^i)(b\,\delta)a$	$\longrightarrow$	$(b\,\delta)(c\,\sigma^i)a$	
σ*-destruction 1*	$(c\,\sigma^i)\mathtt{i}$	$\longrightarrow$	c	
σ*-destruction 2*	$(c\,\sigma^i)\mathtt{j}$	$\longrightarrow$	$\mathtt{j}$	if $\mathtt{j} \neq \mathtt{i}$

- These rules are enough to prevent confluence. For example:

$$(2\sigma^1)(1\,\delta)1 \to_{\sigma_{0\delta}\text{-}tr} ((2\sigma^1)1\,\delta)1 \to_{\sigma\text{-}dest\,1} (2\,\delta)1$$

$$(2\sigma^1)(1\,\delta)1 \to_{\sigma_{1\delta}\text{-}tr} (1\,\delta)(2\sigma^1)1 \to_{\sigma\text{-}dest\,1} (1\,\delta)2$$

- Kamareddine and Nederpelt gave the σ*-generation* rule:

$$\sigma\text{-}generation \qquad (b\,\delta)(\lambda)a \quad \longrightarrow \quad (b\,\delta)(\lambda)((\varphi^1_0)b\,\sigma^1)a$$

- The above rules lead to loss of PSN:

$$(1\,\delta)(\lambda)(2\,\delta)1 \to_{\sigma-gen} (1\,\delta)(\lambda)((\varphi^1_0)1\,\sigma^1)(2\,\delta)1 \to_{\sigma_{0\delta}-tr}$$
$$(1\,\delta)(\lambda)(((\varphi^1_0)1\,\sigma^1)2\,\delta)1 \to_{\sigma-dest\,2} (1\,\delta)(\lambda)(2\,\delta)1 \to_{\sigma-gen} \cdots$$

- To solve the problem, we change the above rules to:

σ-δ-transition 1	$(c\,\sigma^i)(b\,\delta)a$	$\longrightarrow$	$(c\,\sigma^i)((c\,\sigma^i)b\,\delta)a$
σ-δ-transition 2	$(c\,\sigma^i)(b\,\delta)a$	$\longrightarrow$	$(c\,\sigma^i)(b\,\delta)(c\,\sigma^i)a$
σ-disposal	$(c\,\sigma^i)a$	$\longrightarrow$	a if $\mathtt{i} \notin FV(a)$
new σ-generation	$(b\,\delta)(\lambda)a$	$\longrightarrow$	$(b\,\sigma^1)a$

The λs_L-calculus

σ-generation	$(b\,\delta)(\lambda)a$	$\longrightarrow$	$(b\,\sigma^1)a$
σ-λ-transition	$(b\,\sigma^j)(\lambda)a$	$\longrightarrow$	$(\lambda)(b\,\sigma^{j+1})a$
σ_R-generation	$(c\,\sigma^i)(b\,\delta)a$	$\longrightarrow$	$(c\,\sigma^i_R)((L)(c\,\sigma^i)b\,\delta)a$
σ_R-destruction	$(c\,\sigma^i_R)((L)b\,\delta)a$	$\longrightarrow$	$(b\,\delta)(c\,\sigma^i)a$
σ_L-generation	$(c\,\sigma^i)(b\,\delta)a$	$\longrightarrow$	$(c\,\sigma^i_L)(b\,\delta)(L)(c\,\sigma^i)a$
σ_L-destruction	$(c\,\sigma^i_L)(b\,\delta)(L)a$	$\longrightarrow$	$((c\,\sigma^i)b\,\delta)a$
σ-destruction	$(b\,\sigma^j)\mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \text{if } n > j \\ (\varphi^j_0)b & \text{if } n = j \\ \mathtt{n} & \text{if } n < j \end{cases}$
φ-λ-transition	$(\varphi^i_k)(\lambda)a$	$\longrightarrow$	$(\lambda)(\varphi^i_{k+1})a$
φ-δ-transition	$(\varphi^i_k)(a_1\delta)a_2$	$\longrightarrow$	$((\varphi^i_k)a_1\delta)(\varphi^i_k)a_2$
φ-destruction	$(\varphi^i_k)\mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \leq k \end{cases}$

Properties of σ_L

Theorem 1.

- *The σ_L-calculus is strongly normalising.*
- *The σ_L-calculus is confluent.*

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.