Specification of a type system for mCRL2

Collaborative work with Michel Reniers

Jeroen Keiren

January 13, 2011

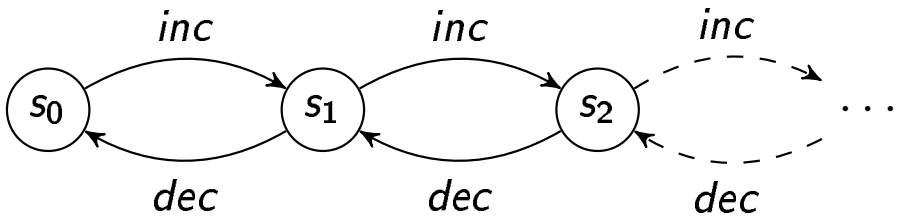
Department of Mathematics and Computer Science

Technische Universiteit Eindhoven University of Technology

Outline

- Specification language
- Process algebra
- Data

$$\begin{array}{lllll} \textbf{proc} & \mathit{Counter}(n{:}\mathbb{N}) & = & & \mathit{inc} \cdot \mathit{Counter}(n+1) \\ & & + & (n > 0) \quad \rightarrow & \mathit{dec} \cdot \mathit{Counter}(n-1) \\ \textbf{init} & \mathit{Counter}(0) & & & \end{array}$$



Types

- Standard data types ($\mathbb{B}, \mathbb{N}^+, \mathbb{N}, \mathbb{Z}, \mathbb{R}$)
- Basic types (S, T, U, $Colour$) (including standard data types)
- Function types ($S \times T \times U \to V$)
- Container types ($List(S)$, $Bag(T)$, $Set(U)$)

Expressions

- Variables (x, y, z)
- Functions $(f, g, h, 0, succ, \triangleright, +)$
- Binding $(\lambda x : S.e,\ \forall x : S.e,\ \exists x : S.e)$
- Application $(e(e_1, \ldots, e_n),\ succ(0),\ 3 \triangleright [5, 481],\ f(x))$

Example (Data specification)

sort $Tree$;

cons $leaf : \mathbb{B} \to Tree$;

$node : (\mathbb{B} \times \mathbb{B} \to \mathbb{B}) \times Tree \times Tree \to Tree$;

map $isNode : Tree \to \mathbb{B}$;

$left, right : Tree \to Tree$;

var $t1, t2 : Tree$;

$b : \mathbb{B}$;

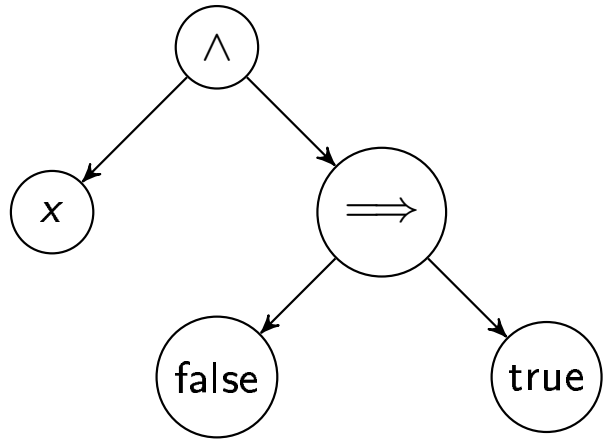
$f : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$;

eqn $isNode(leaf(b)) = \text{false}$;

$isNode(node(f, t1, t2)) = \text{true}$;

$left(node(f, t1, t2)) = t1$;

$right(node(f, t1, t2)) = t2$;



$node(\wedge,$
$\quad leaf(x),$
$\quad node(\implies,$
$\quad\quad leaf(\text{false}),$
$\quad\quad leaf(\text{true})))$

Specifying data, formally

Fix signature $\Sigma = (S_{Basic}, \Omega)$

- S_{Basic} set of basic sorts
- Ω set of function declarations

Note: specifying equations not relevant for typing expressions

Outline

Inferring types of data expressions

Assume:

- Signature Σ
- Context Γ (stack of variable declarations)

$$\frac{x : s \in \Gamma}{\Gamma \vdash_\Sigma x : s}\ (\text{Var}) \qquad \frac{f : s_1 \times \cdots \times s_n \to s \in \Sigma}{\Gamma \vdash_\Sigma f : s_1 \times \cdots \times s_n \to s}\ (\text{Func})$$

$$\frac{\Gamma, x_1 : s_1 \vdash_\Sigma e : s}{\Gamma \vdash_\Sigma (\lambda x_1 : s_1 . e) : s_1 \to s}\ (\text{Abs})$$

$$\frac{\Gamma \vdash_\Sigma e{:}s_1 \times \cdots \times s_n \to s \quad \Gamma \vdash_\Sigma e_1{:}s_1 \quad \cdots \quad \Gamma \vdash_\Sigma e_n{:}s_n}{\Gamma \vdash_\Sigma e(e_1, \ldots, e_n) : s}\ (\text{Appl})$$

Inferring types of data expressions (2)

$$\frac{\Gamma, x_1 : s_1 \vdash_\Sigma e : \mathbb{B}}{\Gamma \vdash_\Sigma (\forall x_1 : s_1 . e) : \mathbb{B}} \text{(Forall)} \qquad \frac{\Gamma, x_1 : s_1 \vdash_\Sigma e : \mathbb{B}}{\Gamma \vdash_\Sigma (\exists x_1 : s_1 . e) : \mathbb{B}} \text{(Exists)}$$

$$\frac{\Gamma, x : s \vdash_\Sigma e : \mathbb{B}}{\Gamma \vdash_\Sigma \{x : s \mid e\} : Set(s)} \text{(Set)} \qquad \frac{\Gamma, x : s \vdash_\Sigma e : \mathbb{N}}{\Gamma \vdash_\Sigma \{x : s \mid e\} : Bag(s)} \text{(Bag)}$$

Outline

mCRL2

Type checking

Overloading & Subtyping

Strict typing

Subtyping

Let:

- $n{:}\mathbb{N}$
- $f{:}\mathbb{Z} \to S$

$\Gamma \vdash_\Sigma f(n) :???$

Two solutions:

1. Require casts: $N2I{:}\mathbb{N} \to \mathbb{Z}$. User writes $f(N2I(n))$
2. Allow for subtyping: $\mathbb{N} \subseteq \mathbb{Z}$

Subtyping (rules)

Add following rule to type inference system:

$$\frac{\Gamma \vdash_\Sigma e : s' \quad s' \subseteq s}{\Gamma \vdash_\Sigma e : s} \text{(Subtyping)}$$

Axioms for $\subseteq$:

$$\frac{}{\mathbb{N}^+ \subseteq \mathbb{N}} \text{(P2N)} \qquad \frac{}{\mathbb{N} \subseteq \mathbb{Z}} \text{(N2I)} \qquad \frac{}{\mathbb{Z} \subseteq \mathbb{R}} \text{(I2R)}$$

$$\frac{\Gamma \vdash_\Sigma s_i' \subseteq s_i}{\Gamma \vdash_\Sigma s_1 \times \cdots \times s_i \times \cdots \times s_n \to s \subseteq s_1 \times \cdots \times s_i' \times \cdots \times s_n \to s}\ (\text{Domain})$$

Let

- $map{:}(\mathbb{N} \to S) \times List(\mathbb{N}) \to List(S)$
- $f{:}\mathbb{R} \to S$
- $x{:}List(\mathbb{N})$

Can $map(f, x)$ be typed?

- $(\mathbb{N} \to S) \times List(\mathbb{N}) \to List(S) \subseteq (\mathbb{N} \to S) \times List(\mathbb{N}) \to List(S)$, if
- $(\mathbb{R} \to S) \subseteq (\mathbb{N} \to S)$, if
- $\mathbb{N} \subseteq \mathbb{R}$

Intuition: function passed as argument to another function can be applied to anything given to it by the context.

$$\frac{\Gamma \vdash_\Sigma s_i \subseteq s_i'}{\Gamma \vdash_\Sigma s_1 \times \cdots \times s_i \times \cdots \times s_n \to s \subseteq s_1 \times \cdots \times s_i' \times \cdots \times s_n \to s}\ (\text{Domain})$$

Let

- $map{:}(\mathbb{R} \to S) \times List(\mathbb{R}) \to List(S)$
- $f{:}\mathbb{N} \to S$
- $x{:}List(\mathbb{R})$

Can $map(f, x)$ be typed?

- $(\mathbb{N} \to S) \times List(\mathbb{R}) \to List(S) \subseteq (\mathbb{R} \to S) \times List(\mathbb{R}) \to List(S)$, if
- $(\mathbb{N} \to S) \subseteq (\mathbb{R} \to S)$, if
- $\mathbb{N} \subseteq \mathbb{R}$.

Problem: how is $f(y)$ defined for $y < 0$?

Subtyping function range

$$\frac{\Gamma \vdash_\Sigma s \subseteq s'}{\Gamma \vdash_\Sigma s_1 \times \cdots \times s_n \to s \subseteq s_1 \times \cdots \times s_n \to s'}\ (\text{Range})$$

Let

- $map{:}(S \to \mathbb{R}) \times List(S) \to List(\mathbb{R})$
- $f{:}S \to \mathbb{N}$
- $x{:}List(S)$

Can $map(f, x)$ be typed?

Intuition: function passed as argument to another function produces only things that can be handled.

Outline

Overloading

Standard definition of $+$:

$$
\begin{aligned}
+&:\mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+ \\
+&:\mathbb{N} \times \mathbb{N}^+ \to \mathbb{N}^+ \\
+&:\mathbb{N}^+ \times \mathbb{N} \to \mathbb{N}^+ \\
+&:\mathbb{N} \times \mathbb{N} \to \mathbb{N} \\
+&:\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \\
+&:\mathbb{R} \times \mathbb{R} \to \mathbb{R}
\end{aligned}
$$

Need support for overloading of operators

Assigning types to subexpressions

mCRL2 uses rewriting to simplify expressions.

- Equations as rewrite rules
- Does expression match with a rule?

Requires efficient ($\mathcal{O}(1)$) matching

$\Downarrow$

Attribute each subexpression with its type

Questions

- Is expression typable?
- What is "the type" of an expression?
- How is the type computed?

Type each subexpression

Goal: assign unique type to each subexpression

Idea: order type deductions $\rightarrow$ prevent/postpone typecasts

First: order deduction rules { Var, Func, Abs, Appl, Forall, Exists, Set, Bag }

$<$

{ Subtyping }

$<$

{ P2N, N2I, I2R, SubList, SubSet, SubBag }

$<$

{ Range }

$<$

{ Domain }

$<$

{ Trans }

Choosing a deduction

$$\dfrac{\dfrac{}{P1}\,(d_1') \quad \cdots \quad \dfrac{}{PN}\,(d_N')}{C1}\,(\text{Rule}_1) \qquad \text{Deduction 1}$$

$$\dfrac{\dfrac{}{Q1}\,(d_1'') \quad \cdots \quad \dfrac{}{QN}\,(d_N'')}{C2}\,(\text{Rule}_2) \qquad \text{Deduction 2}$$

Deduction 1 better than ($<$) Deduction 2,

- if $Rule_1 < Rule_2$, or
- $Rule_1 = Rule_2$, and $d_i' \leq d_i''$ for all i, and $d_i' < d_i''$ for some i

Choosing a deduction (example)

Let $x{:}\mathbb{N}^+$, and

$$f : \mathbb{N}^+ \to S$$
$$f : \mathbb{N} \to S$$

Type $f(x)$

$$\dfrac{\dfrac{f{:}\mathbb{N}^+ \to S \in \Sigma}{\Gamma \vdash_\Sigma f{:}\mathbb{N}^+ \to S}\,(\text{Func}) \qquad \dfrac{x{:}\mathbb{N}^+ \in \Gamma}{\Gamma \vdash_\Sigma x{:}\mathbb{N}^+}\,(\text{Var})}{\Gamma \vdash_\Sigma f(x){:}S}\,(\text{Appl})$$

$$\dfrac{\dfrac{f{:}\mathbb{N} \to S \in \Sigma}{\Gamma \vdash_\Sigma f{:}\mathbb{N} \to S}\,(\text{Func}) \qquad \dfrac{\dfrac{x{:}\mathbb{N}^+ \in \Gamma}{\Gamma \vdash_\Sigma x{:}\mathbb{N}^+}\,(\text{Var}) \qquad \dfrac{}{\mathbb{N}^+ \subseteq \mathbb{N}}\,(\text{P2N})}{\Gamma \vdash_\Sigma x{:}\mathbb{N}}\,(\text{Subtyping})}{\Gamma \vdash_\Sigma f(x){:}S}\,(\text{Appl})$$

Conclusions

- Rules for typing mCRL2
- Determine "the type" of an expression
- Assign types to subexpressions

Future work:

- Make widening explicit (automatically add casts)
- Give algorithm to type check mCRL2
- Investigate generalisation to other languages