9.2.1.

FROM TYPE INFERENCE TO TYPE VERIFICATION IN LOGIC PROGRAMMING

A SIMPLE DOMAIN OF TYPES FOR LOGIC PROGRAMS

[Codish & Lagoon, TCS, 2000]

- concrete terms are abstracted to type terms

- concrete terms are made out of
  - numeric constants
  - variables (capital letters)
  - lists : [ ] , $[t_1 | t_2]$
  - trees : void , tree (t_1, t_2, t_3)

    (trees are representative of generic recursive type constructors)

- type terms are (associative, commutative, idempotent) terms built using
  - a binary set constructor +
  - a collection of monomorphic and polymorphic description symbols
    - monomorphic symbols num, nil, void
    - polymorphic symbols list(–), tree (–)

- the "abstraction function" τ : concrete term → type term

  (to be extended to the real α : $\wp$(concrete terms) → types terms

$$
\tau(t) = \begin{cases}
X & \text{if } t \text{ is the variable } X \\
num & \text{if } t \text{ is a number} \\
nil & \text{if } t = [\,] \\
list(\tau(t_1)) + \tau(t_2) & \text{if } t = [t_1 | t_2] \\
void & \text{if } t = void \\
tree(\tau(t_1)) + \tau(t_2) + \tau(t_3) & \text{if } t = tree(t_1, t_2, t_3)
\end{cases}
$$

- partial order is essentially set inclusion

EXAMPLES OF TYPE ABSTRACTIONS

$\tau([-3,0,7]) = list(\tau(-3)) + \tau([0,7]) =$

$list(num) + list(\tau(0)) + \tau([7]) =$

$list(num) + list(num) + list(\tau(7)) + \tau([\,]) =$

$list(num) + list(num) + nil =$

$list(num) + nil$

$\tau([X,Y]) = list(X) + list(Y) + nil$

$\tau(tree(2, void, void)) = tree(num) + void$

SUCCESS CORRECTNESS WRT TYPES

- the abstract semantic evaluation function for types

$$T_P^{\tau}(I) = \lambda p(\tilde{x}).$$

$$\left\{ p(\tau(\tilde{t}))\mu \;\middle|\; \begin{array}{l} p(\tilde{t}) \leftarrow p_1(\tilde{t}_1), \ldots, p_n(\tilde{t}_n) \in P \\ T_i \in I(p_i(\tilde{x}_i)) \\ \mu \in cU_{ACI}((\tau(\tilde{t}_1), \ldots, \tau(\tilde{t}_n)), (T_1, \ldots, T_n)) \end{array} \right\}$$

cU_{ACI} is the ACI-unification procedure.

- given a specification S_τ, the partial correctness condition is, for each clause c,

$$T_{\{c\}}^{\tau}(S_\tau) \subseteq S_\tau$$

AN EXAMPLE

c1 fib(0,0).

c2 fib(1,1) errore! (colonna errore N2)

c3 fib(X2,N) :- X1 is X2-1, fib(X1,N1),
X0 is X2-2, fib(X0,N0), N2 is N1+N0

- the specification: $S_\tau = fib(X,Y) \mapsto \{fib(num,num)\}$

$T^{\tau}_{\{c_1\}}(S_\tau) = \{fib(num,num)\}$ ok

$T^{\tau}_{\{c_2\}}(S_\tau) = \{fib(num,num)\}$ ok

$T^{\tau}_{\{c_3\}}(S_\tau) = \{fib(num,U)\}$ it might be an error (as is indeed the case)

- if we replace N by N2 we succeed in proving the verification condition

- a "concrete semantics" error often shows up as a type error even with simple "first-order" types

$T^{\tau}_{\{c\}}(I) = \lambda\, p(\tilde{x}).$

$\{ p(\tau(\tilde{t}))\mu \mid \quad c = p(\tilde{t}) \leftarrow p_1(\tilde{t}_1), \dots, p_n(\tilde{t}_n)$

$T_i \in I(p_i(\tilde{x}_i))$

$\mu \in cUAci\, ((\tau(\tilde{t}_1), \dots, \tau(\tilde{t}_n)), (T_1, \dots, T_n)) \}$

EXAMPLE 2

c_1: append([], Xs, Xs).

c_2: append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).

the specification:

$$\mathcal{S}_{\tau} = \text{append}(X,Y,Z) \mapsto \{\text{append}(\text{nil}, \text{nil}+\text{list}(\tau), \text{nil}+\text{list}(\tau)),\ \text{append}(\text{nil}+\text{list}(\tau), \text{nil}+\text{list}(\tau), \text{nil}+\text{list}(\tau))\}$$

$$T^{\tau}_{\{c_2\}}(\mathcal{S}_{\tau}) \not\subseteq \mathcal{S}_{\tau}$$

- the variable Xs is not necessarily a list.

$$T^{\tau}_{\{c_1\}}(\mathcal{S}) = \lambda\, p(\tilde{x}).\ \{\, p(\tau(\tilde{t}))\,\}$$

no clause body
(no use of the specification)

$$\Downarrow$$

append(nil, Xs, Xs)

I/O CORRECTNESS ON TYPES

- a specification is a pair of "type interpretations"

$$\Big(\underset{\substack{\uparrow \\ \text{input}}}{S_\tau^{I}} \;,\; \underset{\substack{\uparrow \\ \text{output}}}{S_\tau^{O}} \Big)$$

- if we expand the "most adequate" fixpoint semantics using the abstraction on types we get the following partial correctness condition

PARTIAL I/O CORRECTNESS CONDITION

1. "unify" the clause head with the input specification

$$\Theta = \{ \mu \mid A \in S_\tau^I(p(\tilde{x})), \mu \in cU_{ACI}(A, p(\tau(\tilde{t}))) \}$$

2. abstract semantics of the procedure calls

(only for those procedure calls which do satisfy the input spec)

$$T_j = \begin{cases} S_\tau^O(p_j(\bar{x}_j)) & \text{if } p_j(\bar{x}_j)\Theta \subseteq S_\tau^I(p_j(\bar{x}_j)) \\ \top & \text{otherwise} \end{cases}$$

3. expanded condition

$$\boxed{\begin{aligned} &\{ p(\tau(\tilde{t}))\mu \mid A_j \in T_j, A \in p(\tilde{x})\Theta, \\ &\qquad \mu \in cU_{ACI}((A, A_1, \ldots, A_n), (p(\tau(\hat{t})), p_1(\tau(\hat{t}_1)), \ldots, p_n(\tau(\hat{t}_n)))) \} \\ &\subseteq S_\tau^O(p(\tilde{x})) \end{aligned}}$$

clause $\quad p(\tilde{t}) \leftarrow p_1(\tilde{t}_1), \ldots, p_n(\tilde{t}_n)$

PARTIAL I/O CORRECTNESS CONDITION

1. ~~...~~ "unify" the clause head with the input specification

$$\Theta = \{ \mu \mid A \in S_P^I(p(\tilde{x})), \mu \in cU_{ACI}(A, p(\tau(\tilde{t}))) \}$$

2. abstract semantics of the procedure calls
(only for those procedure calls which do satisfy the input spec)

$$T_j = \begin{cases} S_P^O(p_j(\bar{x}_j)) & \text{if } p_j(\bar{x}_j)\Theta \subseteq S_P^I(p_j(\bar{x}_j)) \\ T & \text{otherwise} \end{cases}$$

3. expanded condition

$$\boxed{\begin{array}{l} \{ p(\tau(\tilde{t}))\mu \mid A_j \in T_j,\ A \in p(\tilde{x})\Theta, \\ \quad \mu \in cU_{ACI}((A, A_1, \dots, A_n), (p(\tau(\tilde{t})), p_1(\tau(\tilde{t}_1)), \dots, p_n(\tau(\tilde{t}_n)))) \} \\ \subseteq S_P^O(p(\tilde{x})) \end{array}}$$

the first clause of append can now be proved I/O correct w.r.t. ~~...~~ an input specification

$$\{ append(nil + list(T), nil + list(T), U), append(nil, nil + list(T), U) \}$$

$$\mu = \{ Xs \leftarrow nil + list(T) \}$$

clause $p(\tilde{t}) \leftarrow p_1(\tilde{t}_1), \dots, p_n(\tilde{t}_n)$

I/O AND CALL CORRECTNESS WRT TYPES

- Specifications are still pairs of type preconditions and postconditions
    - The choice of the semantics guarantees that preconditions are always satisfied if we prove the sufficient condition.

PARTIAL I/O AND CALL CORRECTNESS CONDITIONS

1. "unify" the clause head with the input spec

$$\Theta = \{\mu \mid A \in S_\pi^I(p(\tilde{x})),\ \mu \in cU_{Acl}(A, p(\tau(H)))\}$$

2. "output" correctness

rather than T_j

$$\{p(\tau(\tilde{f}))\mu \mid A_j \in S_\pi^O(p_j(\bar{x}_j)),\ A \in p(\bar{x})\Theta,$$
$$\mu \in cU_{Acl}((A, A_1, \ldots, A_n)(p(\tau(\bar{f})), p_1(\tau(\bar{f}_1)), \ldots, p_n(\tau(\bar{f}_n)))\}$$
$$\subseteq S_\pi^O(p(\bar{x}))$$

3. "call" correctness

$$\{p_j(\tau(\tilde{t}_j))\mu \mid A_k \in S_\pi^O(p_k(\bar{x}_k)),\ A \in p(\bar{x})\Theta,$$
$$\mu \in cU_{Acl}((A, A_1, \ldots, A_{j-1}),$$
$$(p(\tau(\bar{f})), p_1(\tau(\bar{f}_1)), \ldots, p_{j-1}(\tau(\bar{t}_{j-1}))))\}$$
$$\subseteq S_\pi^I(p_j(\bar{x}_j))$$

all the procedure calls preceding $p_j(\tilde{f}_j)$ do satisfy the output spec).

clause $\quad p(\tilde{f}) \leftarrow p_1(\tilde{f}_1), \ldots, p_n(\bar{f}_n)$