Programming Languages

2nd edition

Tucker and Noonan

Chapter 6
Type Systems

I was eventually persuaded of the need to design programming notations so as to maximize the number of errors that cannot be made, or if made, can be reliably detected at compile time.

C.A.R Hoare

Contents

6.3 Formalizing the Clite Type System

Type map:

$$tm = \{< v_1, t_1 >, < v_2, t_2 >, ..., < v_n, t_n >\}$$

Created by:
(Type Rule 6.1)

$$typing : Declarations \rightarrow TypeMap$$

$$typing(d) = \bigcup_{i \in \{1,...,n\}} < d_i.v, d_i.t >$$

Validity of Declarations:
(Type Rule 6.2)

$$V : Declarations \rightarrow \mathrm{B}$$

$$V(d) = \forall i, j \in \{1,...,n\}(i \neq j \Rightarrow d_i.v \neq d_j.v)$$

Validity of a Clite Program

(Type Rule 6.3)

$$V : Program \rightarrow \mathrm{B}$$
$$V(p) = V(p.\mathrm{decpart}) \wedge V(p.\mathrm{body},\ typing(p.\mathrm{decpart}))$$

Validity of a Clite Statement

(Type Rule 6.4, simplified version for an *Assignment*)

$$V : Statement \times TypeMap \rightarrow \mathrm{B}$$

$$V(s, tm) = true \qquad \text{if } s \text{ is a } Skip$$

$$= s.\text{target} \in tm \wedge V(s.\text{source}, tm) \wedge typeOf(s.\text{target}, tm) = typeOf(s.\text{source}, tm) \qquad \text{if } s \text{ is an } Assignment$$

$$= V(s.\text{test}, tm) \wedge typeOf(s.\text{test}, tm) = bool \wedge V(s.\text{thenbranch}, tm) \wedge V(s.\text{elsebranch}, tm) \qquad \text{if } s \text{ is a } Conditional$$

$$= V(s.\text{test}, tm) \wedge typeOf(s.\text{test}, tm) = bool \wedge V(s.\text{body}, tm) \qquad \text{if } s \text{ is a } Loop$$

$$= V(b_1, tm) \wedge V(b_2, tm) \wedge \ldots \wedge V(b_n, tm) \qquad \text{if } s \text{ is a } Block$$

Validity of a Clite Expression

(Type Rule 6.5, abbreviated versions for *Binary* and *Unary*)

$$V : Expression \times TypeMap \rightarrow \mathrm{B}$$

$$V(e, tm) = true \qquad \text{if } e \text{ is a } Value$$

$$= e \in tm \qquad \text{if } e \text{ is a } Variable$$

$$= V(e.\mathrm{term1}, tm) \wedge V(e.\mathrm{term2}, tm) \wedge \qquad \text{if } e \text{ is a } Binary \wedge$$
$$typeOf(e.\mathrm{term1}, tm) \in \{ float, \mathrm{int} \} \wedge \qquad e.\mathrm{op} \in ArithmeticOp \cup$$
$$typeOf(e.\mathrm{term2}, tm) \in \{ float, \mathrm{int} \} \wedge \qquad \mathrm{Re}\, lationalOp$$
$$typeOf(e.\mathrm{term1}, tm) = typeOf(e.\mathrm{term2}, tm)$$

$$= V(e.\mathrm{term}, tm) \wedge e.\mathrm{op} = \ ! \ \wedge \qquad \text{if } e \text{ is a } Unary$$
$$typeOf(e.\mathrm{term}, tm) = bool$$

Type of a Clite Expression

(Type Rule 6.6, abbreviated version)

$$typeOf : Expression \times TypeMap \rightarrow Type$$

$$typeOf(e, tm) = e.type \quad \text{if } e \text{ is a } Value$$

$$= e.type \quad \text{if } e \text{ is a } Variable \wedge e \in tm$$

$$= typeOf(e.\text{term1}, tm) \quad \text{if } e \text{ is a } Binary \wedge e.\text{op} \in ArithmeticOp$$
$$= boolean \quad \text{if } e \text{ is a } Binary \wedge e.\text{op} \notin ArithmeticOp$$

$$= typeOf(e.\text{term}, tm) \quad \text{if } e \text{ is a } Unary \wedge e.\text{op} = -$$
$$= boolean \quad \text{if } e \text{ is a } Unary \wedge e.\text{op} = \ !$$