Boolean Lang $\mathbb{B}$ (TAPL, Fig 3-1, p 34)

Syntax :

$t ::=$ terms

true

false

if t then t else t

$v ::=$

true

false

Reduction $\boxed{t \longrightarrow t'}$

$$\frac{}{\text{if true then } t_2 \text{ else } t_3 \longrightarrow t_2} \quad \text{E-IFTRUE}$$

$$\frac{}{\text{if false } t_2\ t_3 \longrightarrow t_3} \quad \text{E-IF FALSE}$$

$$\frac{t_1 \longrightarrow t_1'}{\text{if } t_1\ t_2\ t_3 \longrightarrow \text{if } t_1'\ t_2\ t_3} \quad \text{E-IF}$$

- Determinacy ✓
- Uniqueness of N.F : ✓
- Termination : ✓
- Finality : ✓

Arithmetic Exp. (NB) (Fig 3-2, pp 41, TAPL)

IN (Untyped) Extends B (3-1)

t ::= ...
| 0
| succ t
| pred t
| 0? t

nV ::= numeric value
0 zero val
succ nV successor val

New reduction rules:

$t \longrightarrow t'$

$$\frac{t_1 \longrightarrow t_1'}{succ\ t_1 \longrightarrow succ\ t_1'}$$

$$\frac{}{pred\ 0 \longrightarrow 0}$$

$$\frac{}{pred\ (succ\ nv) \longrightarrow nv}$$

$$\frac{t_1 \longrightarrow t_1'}{pred\ t_1 \longrightarrow pred\ t_1'}$$

$$\frac{}{0?\ 0 \longrightarrow true}$$

$$0?\ (succ\ nv) \longrightarrow false$$

$$\frac{t_1 \longrightarrow t_1'}{0?\ t_1 \longrightarrow 0?\ t_1'}$$

- Determinacy: ✓
- Uniqueness of N.F: ✓
- TERMINATION: ✓
- FINALITY of VALUES ✓

But:

Not all normal ~~forms~~ are values!

Defn t is <u>stuck</u> if $t \not\rightarrow$ and t is not in normal form.

stuck ≈ "meaningless state"

Ch 8 TAPL [TAPL]

Typing rules for booleans (B) [Fig 8-1, p 93 TAPL]

B (typed) Extends B (3-1)

New Syntactic Forms

T ::= types

Bool

New typing rules

$t : T$

$$\frac{}{\text{true} : \text{Bool}}$$ T-TRUE

$$\frac{}{\text{false} : \text{Bool}}$$ T-FALSE

$$\frac{t_1 : \text{Bool} \quad t_2 : T \quad t_3 : T}{\text{if } t_1 \; t_2 \; t_3 : T}$$ T-IF

Typing rules for Arithmetic (Fig 8-2, p 93)

$t : T$

T ::= ...

Nat

$$\frac{}{0 : \text{Nat}}$$ T-ZERO

$$\frac{t : \text{Nat}}{\text{succ } t : \text{Nat}}$$ T-SUCC

$$\frac{t : \text{Nat}}{\text{pred } t : \text{Nat}}$$ T-PRED

$$\frac{t : \text{Nat}}{0?\ t : \text{Bool}}$$ T-0?

$t : T$ t is well-typed as T. (or has type T).

t is well-typed if $\exists T : t : T$.

- Uniqueness: $t : T_1$ and $t : T_2 \Rightarrow T_1 = T_2$

SAFETY = PROGRESS + PRESERVATION

Progress : A well-typed term is not stuck.
(either it is a value. or it reduces)

Safety : If t is well-typed and $t \longrightarrow t'$,
then t is well-typed.

For typed Arith Exp:

progress : ✓

safety : ✓

$$t : T \;,\; t \longrightarrow t' \;\Rightarrow\; t' : T$$

A given term t may exhibit the following three types of behaviour:

1. It reduces to another term t'
   $t \longrightarrow t'$

2. It does not reduce: Here there are three cases:
   a) t is a value
   b) t (is not a value and) is not well-typed either
   c) t (is not a value, but) is well-typed. (Runtime errors)

Example of (c) is division by 0.
eg 5/0.

In designing programming language type systems, we have 2 design choices.

1) We design a type system so that no runtime errors occur. In the above example, we will then need to design a type system that statically flags divide by zero errors. The type checker will need to verify an undecidable property (div by 0).

2) We allow for runtime errors, so a term type checked by the type system may indeed be irreducible. It is then flagged as a runtime error (or exception)

The second option is the more reasonable one.

Note that one of the important benefits of Type checking is that types of arguments of some primitives need not be checked at runtime.

Eg in plus t_1 t_2, a type checker will guarantee that plus will receive numerical arguments always.

If $t_1 : Nat$ & $t_2 : Nat$, and if

$t_1 \xrightarrow{*} v_1$ then $v_1 : Nat$

$t_2 \xrightarrow{*} v_2$ then $v_2 : Nat$,

so plus t_1 $t_2 \xrightarrow{*}$ plus v_1 v_2

plus when applied will always receive two values, each of type Nat.

$$\frac{}{\text{true} \quad \text{bool-val}} \qquad \frac{}{0 \quad \text{nat-val}}$$

$$\frac{}{\text{false} \quad \text{bool-val}} \qquad \frac{t \quad \text{nat-val}}{\text{succ } t \quad \text{nat-val}}$$

$$\frac{t \quad \text{bool-val}}{t \quad \text{val}} \qquad \frac{t \quad \text{nat-val}}{t \quad \text{val}}$$

Terms

$t ::= \text{true} \mid \text{false} \mid \text{if } t\ t\ t \mid 0 \mid \text{succ } t \mid \text{pred } t \mid 0?\ t$

Dynamics:

$$\frac{t_1 \quad \text{bool-val} \qquad t_1 \equiv \text{true}}{\text{if } t_1\ t_2\ t_3 \longrightarrow t_2} \qquad \text{E-IFTRUE}$$

$$\frac{t_1 \quad \text{bool-val} \qquad t_1 \equiv \text{false}}{\text{if } t_1\ t_2\ t_3 \longrightarrow t_3} \qquad \text{E-IFFALSE}$$

$$\frac{t_1 \longrightarrow t_1'}{\text{succ } t_1 \longrightarrow \text{succ } t_2} \qquad \text{E-SUCC}$$

$$\frac{t \quad \text{nat-val} \qquad t \equiv \text{succ } t_1' \qquad t' \quad \text{nat-val} \qquad t \neq 0}{\text{pred } t_1 \longrightarrow t_1'} \qquad \text{E-PREDSUCC}$$

$$\frac{t_1 \longrightarrow t_1'}{\text{pred } t_1 \longrightarrow \text{pred } t_1'} \quad \text{E-PRED}$$

$$\frac{t_1 \text{ nat-val} \qquad t_1 \equiv 0}{\text{o? } t_1 \longrightarrow \text{true}} \quad \text{E-ISZEROZERO}$$

$$\frac{t_1 \text{ nat-val} \qquad t \equiv \text{succ } t_1}{\text{o? } t_1 \longrightarrow \text{false}} \quad \text{E-ISZEROSUCC}$$

$$\frac{t_1 \longrightarrow t_1'}{\text{o? } t_1 \longrightarrow \text{o? } t_1'} \quad \text{E-ISZERO}$$

RUNTIME CHECKS are done at the time of reducing via ELIMINATION rules.

RUNTIME TYPE CHECKS
- t nat-val
- t bool-val

] Vidahan's RUNTIME TYPE-ERROR

RUNTIME (BOUNDS) ERROR if none of the bound checks are true

RUNTIME BOUNDS CHECK $t \equiv \text{succ } t'$ where t' nat-val
ie $t \neq 0$

A type system, if well-designed, could allow the RUNTIME TYPE CHECKS to be eliminated because the type system guarantees that RUNTIME TYPE-ERRORS cannot occur (PROGRESS PROPERTY).

	COMPILE TIME	RUNTIME
TYPE CHECK	YES	NO
BOUNDS CHECK	NO	YES

Modeling runtime errors

$$\frac{}{\text{pred } 0 \quad \text{err}}$$

$$\frac{t_1 \quad \text{err}}{\text{succ } t_1 \quad \text{err}}$$

$$\frac{t_1 \quad \text{err}}{\text{pred } t_1 \quad \text{err}}$$

$$\frac{t_1 \quad \text{err}}{\text{if } t_1 \; t_2 \; t_3 \quad \text{err}}$$

$$\frac{}{0 \quad \text{val}}$$

$$\frac{t_1 \quad \text{val}}{\text{succ } t_1 \quad \text{val}}$$

$$\frac{}{\text{true} \quad \text{val}}$$

$$\frac{}{\text{false} \quad \text{val}}$$

$$\frac{t_1 \quad \text{err}}{0?\ t_1 \quad \text{err}}$$

Lemma: if $\vdash t$ val then $\not\vdash t$ err
if $\vdash t$ err then $\not\vdash t$ val

Thm: if $\vdash t$ err then
t is in normal form.

Progress: If $\vdash t : T$, then exactly one of three are true.

1) $\vdash t$ val [RESULT]
2) $\vdash t$ err [RUNTIME EXCEPTION/ERROR]
3) $\vdash t \longrightarrow t'$ for some t' [PGM that can run]
such $t' : T$

Corollary:

If $\vdash t : T$ and $t \xrightarrow{*} t'$, then

i) $\vdash t$ val and $t \not\rightarrow$ (normal form)

ii) $\vdash t$ err and $t \not\rightarrow$

iii) $\vdash t' \longrightarrow t''$ & $\underbrace{\vdash t'' : T}_{\text{preservation}}$ for some t''.

UNTYPED λ-CALCULUS [Ch 5, TAPL]

Syntax

$t ::=$ terms

x	var
$\lambda x.t$	abs
$t\ t$	app

$v ::=$ values: abstraction val

$\lambda x.t$

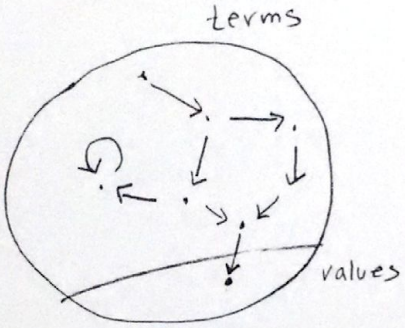
Fig 5.3 TAPL, p72

1-step Reduction (CBvalue)

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{E-APP}_1$$

$$\frac{t_2 \longrightarrow t_2'}{t_1\ t_2 \longrightarrow t_1\ t_2'} \quad \text{E-APP}_2$$

$$(\lambda x.t)\ v \longrightarrow [x/v]t \quad [\text{E-APPABS}]$$



A term is final or irreducible in normal form if it cannot be reduced.

$t \xrightarrow{*} t$ is the reflexive, transitive closure of $\longrightarrow$.

Evaluation is a relation between terms & values.
A term t "evaluates to a value v, if $t \xrightarrow{*} v$

- Determinacy: If $t \longrightarrow t_1$ & $t \longrightarrow t_2$, is $t_1 = t_2$?
- Termination: Does every sequence of reductions terminate? i.e., is $\longrightarrow$ well-founded?
- Finality of values: Are all values final?

For untyped λ-calculus (CBV)

1) Determinacy: no (but confluence YES)
2) UNIQUENESS of N.F.
3) Termination: no
4) Finality: YES

Ch 9 TAPL

$t \longrightarrow t'$

Fig 9-1 Pure STLC

$t ::= x \mid \lambda x:T.\, t \mid t\ t$

$v ::= \lambda x:T.\, t$

$T ::= T \to T$

$\Gamma ::= \emptyset \mid \Gamma, x:T \quad (x \notin dom(\Gamma))$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'}$$

$$(\lambda x:T.\, t)\, v \longrightarrow [x \mapsto v]\, t$$

Typing $\Gamma \vdash t : T$

$$\frac{x:T \in \Gamma}{\Gamma \vdash x : T}\ \text{T-VAR}$$

$$\frac{\Gamma, x:T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x:T_1.\, t_2 : T_1 \to T_2}\ \text{T-VAR}$$

$$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2}\ \text{T-APP}$$

No term in Pure STLC is well-typed!

∴ add type variables to the set of Types

so STLC with type vars.

STLC (2)

Uniqueness: Given Γ,

$\Gamma, t : T_1$ & $\Gamma, t : T_2 \Rightarrow T_1 = T_2$

TYPING & CANONICAL FORMS

1) $v : Bool \Rightarrow v = True$ or $v = False$

2) $v : T_1 \to T_2 \Rightarrow v = \lambda x : T_1 . t$

PROGRESS :

PRESERVATION