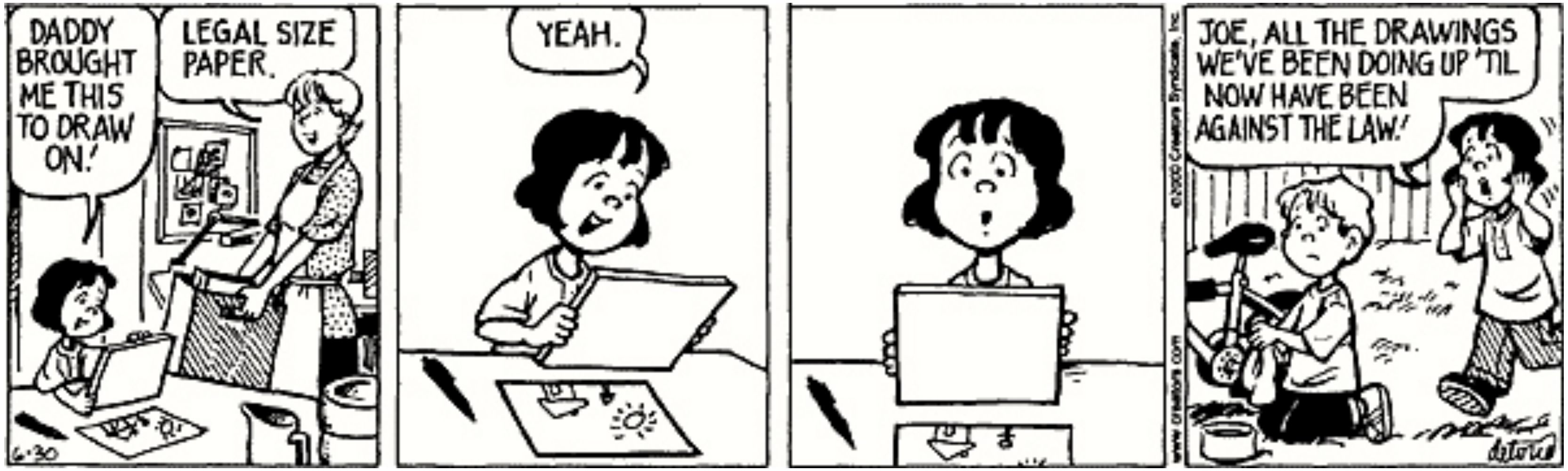
CS 320 : Formal Semantics



Marco Gaboardi
CDS 1019
gaboardi@bu.edu

Operational Semantics

- It is usually provided at a level of abstraction that is independent from the machine.
- The detailed characteristics of the particular computer would make actions difficult to describe/understand.
- Different formalism has been developed to describe the operational semantics in a machine-independent way.

We will look into formal rules and derivations.

Language for the Interpreter (simplified)

The language for the interpreter can be described by the following grammar:

```
<const> ::= int | name
<prog>  ::= quit | <com> ; <prog>
<com>   ::= push <const> | pop | add | sub | mul | div
```

- A program is a sequence of commands followed by `quit`.
- A command is one the keywords above - in the case of push this is followed by a constant.
- A (simplified) constant is either an int or a string.
- We will denote arbitrary programs with `p,p',`...

Operational semantics for the interpreter

$$(\mathtt{p/S}) \rightarrow (\mathtt{p'/S'})$$

Here `(p/S)` is a configuration where p is a program and S is a stack. We call these pairs configurations because we think in terms of an "abstract machine".

We can think about the stack as a list of values (denoted with `v`):

$$\mathtt{v_n::\ldots::v_2::v_1::[]}$$

We say that from the configuration (p/S) we can step (or reduce) to the configuration (p'/S') in one step.

Summary of some rules:

Let's give to each rule a name.

(A) (push num;p/S) $\longrightarrow$ (p/num::S)

(B) (add;p/int(v_2)::int(v_1)::S) $\longrightarrow$ (p/int(v_2+v_1)::S)

(C) (add;p/v_1::[]) $\longrightarrow$ Error

(D) (add;p\[]) $\longrightarrow$ Error

(E) (add;p/name(v)::S) $\longrightarrow$ Error

(F) (add;p/int(v_1)::name(v_2)::S) $\longrightarrow$ Error

Important: the rules do not have an order. We need to design them so that it is clear which one we can apply at each step.

Multiple steps of Operational semantics

We can be more precise and define a multistep semantics as:

$$\boxed{\texttt{(p,S)} \rightarrow^{k} \texttt{(p',S')}}$$

We say that from the configuration (p, S) we can step (or reduce) to the configuration (p',S') in k steps.

$$\frac{}{\texttt{(p/S)} \rightarrow 0 \ \texttt{(p/S)}}$$

$$\frac{\texttt{(p/S)} \rightarrow \texttt{(p'/S')} \qquad \texttt{(p'/S')} \rightarrow k \ \texttt{(p''/S'')}}{\texttt{(p/S)} \rightarrow k+1 \ \texttt{(p''/S'')}}$$

An example:

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{(\texttt{add/int(4)::int(5)::S}) \to (\texttt{/int(4+5)::S})}
      \qquad
      \dfrac{}{(\texttt{/int(4+5)::S}) \to^{0} (\texttt{/int(4+5)::S})}
    }{
      \dfrac{}{(\texttt{push 4;add/int(5)::S}) \to (\texttt{add/int(4)::int(5)::S})}
      \qquad
      (\texttt{add/int(4)::int(5)::S}) \to^{1} (\texttt{/int(4+5)::S})
    }
  }{
    \dfrac{}{(\texttt{push 5;push 4;add/S}) \to (\texttt{push 4;add/int(5)::S})}
    \qquad
    (\texttt{push 4;add/int(5)::S}) \to^{2} (\texttt{/int(4+5)::S})
  }
}{
  (\texttt{push 5;push 4;add/S}) \to^{3} (\texttt{/int(9)::S})
}
$$

An example:

`(push 5;push 4;add/S)` $\longrightarrow^3$ `(/int(9)::S)`

An example:

$$\frac{}{(\texttt{push 5;push 4;add}/S) \rightarrow^3 (/\texttt{int(9)::}S)}$$

An example:

$$\frac{\overline{(\texttt{push 5;push 4;add}/\texttt{S}) \;\longrightarrow\; (\texttt{push 4;add}/\texttt{int(5)::S})}}{(\texttt{push 5;push 4;add}/\texttt{S}) \;\longrightarrow 3\; (/\texttt{int(9)::S})}$$

An example:

$$\dfrac{\dfrac{}{(\texttt{push 5;push 4;add}/\texttt{S}) \rightarrow (\texttt{push 4;add}/\texttt{int(5)::S})} \qquad \dfrac{}{(\texttt{push 4;add}/\texttt{int(5)::S}) \rightarrow^{2} (/\texttt{int(4+5)::S})}}{(\texttt{push 5;push 4;add}/\texttt{S}) \rightarrow^{3} (/\texttt{int(9)::S})}$$

An example:

$$\dfrac{\dfrac{}{\texttt{(push 5;push 4;add/S)} \to \texttt{(push 4;add/int(5)::S)}} \qquad \dfrac{\dfrac{}{\texttt{(push 4;add/int(5)::S)} \to \texttt{(add/int(4)::int(5)::S)}}}{\texttt{(push 4;add/int(5)::S)} \to^{2} \texttt{(/int(4+5)::S)}}}{\texttt{(push 5;push 4;add/S)} \to^{3} \texttt{(/int(9)::S)}}$$

An example:

$$\dfrac{\dfrac{}{\texttt{(push 5;push 4;add/S)} \to \texttt{(push 4;add/int(5)::S)}} \qquad \dfrac{\dfrac{}{\texttt{(push 4;add/int(5)::S)} \to \texttt{(add/int(4)::int(5)::S)}} \qquad \dfrac{}{\texttt{(add/int(4)::int(5)::S)} \to 1 \texttt{(/int(4+5)::S)}}}{\texttt{(push 4;add/int(5)::S)} \to 2 \texttt{(/int(4+5)::S)}}}{\texttt{(push 5;push 4;add/S)} \to 3 \texttt{ (/int(9)::S)}}$$

An example:

(add/int(4)::int(5)::S) → (/int(4+5)::S)

(push 4;add/int(5)::S) → (add/int(4)::int(5)::S)   (add/int(4)::int(5)::S) →1 (/int(4+5)::S)

(push 5;push 4;add/S) → (push 4;add/int(5)::S)   (push 4;add/int(5)::S) →2 (/int(4+5)::S)

(push 5;push 4;add/S) →3 (/int(9)::S)

An example:

$$
\dfrac{
\dfrac{}{(\texttt{push 5;push 4;add}/\texttt{S}) \rightarrow (\texttt{push 4;add}/\texttt{int(5)::S})}
\quad
\dfrac{
\dfrac{}{(\texttt{push 4;add}/\texttt{int(5)::S}) \rightarrow (\texttt{add}/\texttt{int(4)::int(5)::S})}
\quad
\dfrac{
\dfrac{}{(\texttt{add}/\texttt{int(4)::int(5)::S}) \rightarrow (/\texttt{int(4+5)::S})}
\quad
\dfrac{}{(/\texttt{int(4+5)::S}) \rightarrow_0 (/\texttt{int(4+5)::S})}
}{(\texttt{add}/\texttt{int(4)::int(5)::S}) \rightarrow_1 (/\texttt{int(4+5)::S})}
}{(\texttt{push 4;add}/\texttt{int(5)::S}) \rightarrow_2 (/\texttt{int(4+5)::S})}
}{(\texttt{push 5;push 4;add}/\texttt{S}) \rightarrow_3 (/\texttt{int(9)::S})}
$$

Language for the Interpreter (simplified)

The language for the interpreter can be described by the following grammar:

```
<const> ::= int | name
<prog>  ::= <com> | <com>;<prog>
<com>   ::= push <const> | pop | add | sub | mul | div
```

What is the form of a program?

```
<com>;<com>;…;<com>
```

Is this the common way we write programs?

Arithmetical expressions: shape of expressions

- Let us consider this simple language for expressions

```
<expr> ::= <expr> <addop> <expr> | nat
<addop>::= add | sub
```

What are the challenges here?

What is the form of a program?

```
nat(add|sub)nat(add|sub)nat...
```

Do we need a stack here?

Operational semantics for basic arithmetical expressions

$$e \longrightarrow e'$$

Here the expression e is itself a configuration. We already have all the information we need to execute it.

Some rules

$$\frac{}{\texttt{v}_1\ \texttt{add}\ \texttt{v}_2 \rightarrow \texttt{v}_1\texttt{+}\texttt{v}_2}$$

$$\frac{}{\texttt{v}_1\ \texttt{sub}\ \texttt{v}_2 \rightarrow \texttt{v}_1\texttt{-}\texttt{v}_2}$$

What can we do when we have expressions instead of a values $\texttt{v}_1$ $\texttt{v}_2$?

Contextual rules

$$\frac{\texttt{e}_1 \longrightarrow \texttt{e}_1'}{\texttt{e}_1 \ \texttt{add} \ \texttt{e}_2 \longrightarrow \texttt{e}_1' \ \texttt{add} \ \texttt{e}_2}$$

We can use the fact that $\texttt{e}_1$ is recursive and hypothetical reasoning.

Contextual rules

$$\frac{\texttt{e}_2 \;\rightarrow\; \texttt{e}_2{}'}{\texttt{v}_1\ \texttt{add}\ \texttt{e}_2 \;\rightarrow\; \texttt{v}_1\ \texttt{add}\ \texttt{e}_2{}'}$$

We can use the fact that e_2 is recursive and hypothetical reasoning.

Summing up

Are we done?

$$\frac{}{v_1 \ \texttt{add} \ v_2 \to v_1 + v_2}$$

$$\frac{}{v_1 \ \texttt{sub} \ v_2 \to v_1 - v_2}$$

$$\frac{e_1 \to e_1'}{e_1 \ \texttt{add} \ e_2 \to e_1' \ \texttt{add} \ e_2}$$

$$\frac{e_2 \to e_2'}{v_1 \ \texttt{add} \ e_2 \to v_1 \ \texttt{add} \ e_2'}$$

$$\frac{e_1 \to e_1'}{e_1 \ \texttt{sub} \ e_2 \to e_1' \ \texttt{sub} \ e_2}$$

$$\frac{e_2 \to e_2'}{v_1 \ \texttt{sub} \ e_2 \to v_1 \ \texttt{sub} \ e_2'}$$

Multiple steps of Operational semantics

We can define a multistep semantics as:

$$\boxed{e \;\to^{k}\; e'}$$

$$\frac{}{e \;\to^{0}\; e}$$

$$\frac{e \;\to\; e' \qquad e' \;\to^{k}\; e''}{e \;\to^{k+1}\; e''}$$

Summing up

$$\frac{}{v_1 \ \mathtt{add} \ v_2 \rightarrow v_1 + v_2}\ (+V)$$

$$\frac{}{v_1 \ \mathtt{sub} \ v_2 \rightarrow v_1 - v_2}\ (-V)$$

$$\frac{e_1 \rightarrow e_1'}{e_1 \ \mathtt{add} \ e_2 \rightarrow e_1' \ \mathtt{add} \ e_2}\ (+e1)$$

$$\frac{e_2 \rightarrow e_2'}{v_1 \ \mathtt{add} \ e_2 \rightarrow v_1 \ \mathtt{add} \ e_2'}\ (+e2)$$

$$\frac{e_1 \rightarrow e_1'}{e_1 \ \mathtt{sub} \ e_2 \rightarrow e_1' \ \mathtt{sub} \ e_2}\ (-e1)$$

$$\frac{e_2 \rightarrow e_2'}{v_1 \ \mathtt{sub} \ e_2 \rightarrow v_1 \ \mathtt{sub} \ e_2'}\ (-e1)$$

$$\frac{}{e \rightarrow^{0} e}\ (s0)$$

$$\frac{e \rightarrow e' \qquad e' \rightarrow^{k} e''}{e \rightarrow^{k+1} e''}\ (s1)$$

An example:

$$
\dfrac{
\dfrac{
\dfrac{}{2\ \texttt{add}\ 3 \longrightarrow 5}\,(\texttt{+v})
}{
2\ \texttt{add}\ 3\ \texttt{add}\ 4 \longrightarrow 5\ \texttt{add}\ 4
}\,(\texttt{+e1})
\qquad
\dfrac{
\dfrac{}{5\ \texttt{add}\ 4 \longrightarrow 9}\,(\texttt{+v})
\quad
\dfrac{}{9 \longrightarrow_0 9}\,(\texttt{s0})
}{
5\ \texttt{add}\ 4 \longrightarrow_1 9
}\,(\texttt{s1})
}{
2\ \texttt{add}\ 3\ \texttt{add}\ 4 \longrightarrow_2 9
}\,(\texttt{s1})
$$

An example:

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$
$$2 \ \texttt{add} \ 3 \ \texttt{add} \ 4 \longrightarrow_2 \quad 9$$

An example:

$$\frac{2\ \texttt{add}\ 3\ \texttt{add}\ 4 \to 5\ \texttt{add}\ 4 \qquad 5\ \texttt{add}\ 4 \to_1 9}{2\ \texttt{add}\ 3\ \texttt{add}\ 4 \to_2 9}\ (\texttt{s1})$$

An example:

$$\dfrac{\dfrac{2\ \mathtt{add}\ 3 \longrightarrow 5}{2\ \mathtt{add}\ 3\ \mathtt{add}\ 4 \longrightarrow 5\ \mathtt{add}\ 4}\,\mathtt{(+e1)} \qquad 5\ \mathtt{add}\ 4 \longrightarrow_1 9}{2\ \mathtt{add}\ 3\ \mathtt{add}\ 4 \longrightarrow_2 9}\,\mathtt{(s1)}$$

An example:

$$\dfrac{\dfrac{\dfrac{}{2\ \texttt{add}\ 3 \ \longrightarrow\ 5}\,\texttt{(+v)}}{2\ \texttt{add}\ 3\ \texttt{add}\ 4 \ \longrightarrow\ 5\ \texttt{add}\ 4}\,\texttt{(+e1)} \qquad\qquad 5\ \texttt{add}\ 4 \ \longrightarrow_1\ 9}{2\ \texttt{add}\ 3\ \texttt{add}\ 4 \ \longrightarrow_2\ 9}\,\texttt{(s1)}$$

An example:

$$\dfrac{\dfrac{\dfrac{}{2 \ \mathtt{add}\ 3 \longrightarrow 5}\,(\mathtt{+v})}{2 \ \mathtt{add}\ 3 \ \mathtt{add}\ 4 \longrightarrow 5 \ \mathtt{add}\ 4}\,(\mathtt{+e1}) \qquad \dfrac{5 \ \mathtt{add}\ 4 \longrightarrow 9 \qquad 9 \longrightarrow_0 9}{5 \ \mathtt{add}\ 4 \longrightarrow_1 9}\,(\mathtt{s1})}{2 \ \mathtt{add}\ 3 \ \mathtt{add}\ 4 \longrightarrow_2 9}\,(\mathtt{s1})$$

An example:

$$\dfrac{\dfrac{\dfrac{}{2\ \mathtt{add}\ 3 \longrightarrow 5}\,(\mathtt{+v})}{2\ \mathtt{add}\ 3\ \mathtt{add}\ 4 \longrightarrow 5\ \mathtt{add}\ 4}\,(\mathtt{+e1}) \qquad \dfrac{\dfrac{}{5\ \mathtt{add}\ 4 \longrightarrow 9}\,(\mathtt{+v}) \quad 9 \longrightarrow_0 9}{5\ \mathtt{add}\ 4 \longrightarrow_1 9}\,(\mathtt{s1})}{2\ \mathtt{add}\ 3\ \mathtt{add}\ 4 \longrightarrow_2 9}\,(\mathtt{s1})$$

An example:

$$
\dfrac{
  \dfrac{
    \dfrac{}{2 \ \mathtt{add}\ 3 \longrightarrow 5}\,(\mathtt{+v})
  }{
    2 \ \mathtt{add}\ 3 \ \mathtt{add}\ 4 \longrightarrow 5 \ \mathtt{add}\ 4
  }\,(\mathtt{+e1})
  \qquad
  \dfrac{
    \dfrac{}{5 \ \mathtt{add}\ 4 \longrightarrow 9}\,(\mathtt{+v})
    \quad
    \dfrac{}{9 \longrightarrow_0 9}\,(\mathtt{s0})
  }{
    5 \ \mathtt{add}\ 4 \longrightarrow_1 9
  }\,(\mathtt{s1})
}{
  2 \ \mathtt{add}\ 3 \ \mathtt{add}\ 4 \longrightarrow_2 9
}\,(\mathtt{s1})
$$

$\boxed{\text{Is this the only derivation?}}$

Another example:

$$
\dfrac{
\dfrac{\dfrac{}{3 \;\mathtt{add}\; 4 \longrightarrow 7}\,(\mathtt{+v})}{2 \;\mathtt{add}\; 3 \;\mathtt{add}\; 4 \longrightarrow 2 \;\mathtt{add}\; 7}\,(\mathtt{+e2})
\qquad
\dfrac{\dfrac{}{2 \;\mathtt{add}\; 7 \longrightarrow 9}\,(\mathtt{+v}) \quad \dfrac{}{9 \longrightarrow_0 9}\,(\mathtt{s0})}{2 \;\mathtt{add}\; 7 \longrightarrow_1 9}\,(\mathtt{s1})
}{2 \;\mathtt{add}\; 3 \;\mathtt{add}\; 4 \longrightarrow_2 9}\,(\mathtt{s1})
$$

Can we decrease the number of rules in our semantics?

Semantics 1

$$\frac{}{v_1 \text{ add } v_2 \rightarrow v_1 + v_2}\ (+V)$$

$$\frac{}{v_1 \text{ sub } v_2 \rightarrow v_1 - v_2}\ (-V)$$

$$\frac{e_2 \rightarrow e_2'}{v_1 \text{ add } e_2 \rightarrow v_1 \text{ add } e_2'}\ (+e)$$

$$\frac{e_2 \rightarrow e_2'}{v_1 \text{ sub } e_2 \rightarrow v_1 \text{ sub } e_2'}\ (-e)$$

$$\frac{}{e \rightarrow^{0} e}\ (s0)$$

$$\frac{e \rightarrow e' \qquad e' \rightarrow^{k} e''}{e \rightarrow^{k+1} e''}\ (s1)$$

Semantics 2

$$\frac{}{v_1 \;\mathtt{add}\; v_2 \rightarrow v_1+v_2}\;(+V)$$

$$\frac{}{v_1 \;\mathtt{sub}\; v_2 \rightarrow v_1-v_2}\;(-V)$$

$$\frac{e_1 \rightarrow e_1'}{e_1 \;\mathtt{add}\; e_2 \rightarrow e_1' \;\mathtt{add}\; e_2}\;(+e)$$

$$\frac{e_1 \rightarrow e_1'}{e_1 \;\mathtt{sub}\; e_2 \rightarrow e_1' \;\mathtt{sub}\; e_2}\;(-e)$$

$$\frac{}{e \rightarrow^{0} e}\;(s0)$$

$$\frac{e \rightarrow e' \qquad e' \rightarrow^{k} e''}{e \rightarrow^{k+1} e''}\;(s1)$$

Grammar vs operational semantics

- We can use the shape of programs to choose the "right" semantics:

```
<expr> ::= nat <addop> <expr> | nat
<addop>::= add | sub
```

$$\frac{}{v_1 \ \mathtt{add}\ v_2 \ \longrightarrow v_1+v_2}\ (\mathtt{+V})$$

$$\frac{}{v_1 \ \mathtt{sub}\ v_2 \ \longrightarrow v_1-v_2}\ (\mathtt{-V})$$

$$\frac{e_2 \ \longrightarrow \ e_2'}{v_1 \ \mathtt{add}\ e_2 \ \longrightarrow v_1 \ \mathtt{add}\ e_2'}\ (\mathtt{+e})$$

$$\frac{e_2 \ \longrightarrow \ e_2'}{v_1 \ \mathtt{sub}\ e_2 \ \longrightarrow v_1 \ \mathtt{sub}\ e_2'}\ (\mathtt{-e})$$

$$\frac{}{e \ \longrightarrow_0 \ e}\ (\mathtt{s0})$$

$$\frac{e \ \longrightarrow \ e' \qquad e' \ \longrightarrow_k \ e''}{e \ \longrightarrow_{k+1} \ e''}\ (\mathtt{s1})$$

Boolean expressions

- Let us consider this simple language for Boolean expressions

```
<bexpr> ::= <const> <bop> <bexpr> | <const>
<bop>::= and | or | eq
<const>::= bool | int
```

What are the challenges here?

Operational semantics for basic boolean expressions

$$\mathtt{e} \longrightarrow \mathtt{?}$$

Here the expression $\mathtt{e}$ is itself a configuration. We already have all the information we need to execute it.

What can $\mathtt{?}$ be?

Operational semantics for basic boolean expressions

$$\texttt{e} \longrightarrow \texttt{?}$$

Here the expression `e` is itself a configuration. We already have all the information we need to execute it.

What can `?` be?

$$\texttt{e} \longrightarrow \texttt{e'} \qquad\qquad \texttt{e} \longrightarrow \texttt{err}$$

Rules

c here is a configuration, either an expression e or err

$$\frac{v_1\ v_2\ \text{different type}}{v_1\ \text{eq}\ v_2 \rightarrow \text{err}}$$

$$\frac{v_1\ v_2\ \text{same type}}{v_1\ \text{eq}\ v_2 \rightarrow v_1 = v_2}$$

$$\frac{v_1\ v_2\ \text{bool}}{v_1\ \text{and}\ v_2 \rightarrow v_1 \wedge v_2}$$

$$\frac{v_1\ v_2\ \text{bool}}{v_1\ \text{or}\ v_2 \rightarrow v_1 \vee v_2}$$

$$\frac{e_1 \rightarrow e_1' \qquad e_1' \neq \text{Err}}{e_1\ \text{bop}\ e_2 \rightarrow e_1'\ \text{bop}\ e_2}$$

$$\frac{e_2 \rightarrow e_2' \qquad e_2' \neq \text{Err}}{v_1\ \text{bop}\ e_2 \rightarrow v_1\ \text{bop}\ e_2'}$$

$$\frac{}{c \rightarrow^0 c}$$

$$\frac{c \rightarrow c' \qquad c' \rightarrow^k c''}{c \rightarrow^{k+1} c''}$$

$$\frac{v_1\ v_2\ \text{not bool}}{v_1\ \text{and}\ v_2 \rightarrow \text{err}}$$

$$\frac{v_1\ v_2\ \text{not bool}}{v_1\ \text{or}\ v_2 \rightarrow \text{err}}$$

$$\frac{e_1 \rightarrow \text{err}}{e_1\ \text{bop}\ e_2 \rightarrow \text{err}}$$

$$\frac{e_2 \rightarrow \text{err}}{v_1\ \text{bop}\ e_2 \rightarrow \text{err}}$$

What can we do to have a more efficient semantics for boolean expressions?

What can we do to have a more efficient semantics for boolean expressions?

What if we know that one of the elements of an or is true or one of the elements of an and is false?

More efficient rules

$$\frac{\texttt{e}_2 \longrightarrow \texttt{e}_2'}{\texttt{v}_1 \texttt{ bop } \texttt{e}_2 \longrightarrow \texttt{v}_1 \texttt{ bop } \texttt{e}_2'}$$

We could change this rule:

$$\frac{}{\texttt{true or } \texttt{e}_2 \longrightarrow \texttt{true}}$$

$$\frac{\texttt{e}_2 \longrightarrow \texttt{e}_2'}{\texttt{false or } \texttt{e}_2 \longrightarrow \texttt{false or } \texttt{e}_2'}$$

Are the two semantics equivalent?

What if we want to check the second branch first?