

Programming Languages, CPSC 604—Slides 3

Small Step Operational Semantics

Jaakko Järvi

January 28, 2009

Outline

- 1 Small-step semantics
- 2 Properties of \mathcal{NB}
- 3 Termination

Big-step vs. small-step semantics

- The relation \Downarrow defines the so called **big-step**, or **natural semantics**
- $t \Downarrow u$ relates a term t to its meaning u (the final result) directly
- Small-step semantics, instead, defines a single evaluation step towards the final result
- Notation for the small-step evaluation relation:

$$t \rightarrow u$$

Think of an abstract machine making one computation step — moving from one state to the next.

- The meaning is then defined as a chain of these evaluations, until the final result.

$$t \rightarrow t_1 \rightarrow t_2 \rightarrow \dots \rightarrow u$$

- If t evaluates to u in zero or more steps, we write:

$$t \twoheadrightarrow u$$

Thus, \twoheadrightarrow is the reflexive and transitive closure of \rightarrow

\mathcal{NB} with small-step semantics

$$\frac{\text{E-Iszero} \quad t \rightarrow t'}{\text{iszero } t \rightarrow \text{iszero } t'}$$

$$\text{E-IszeroZero} \quad \text{iszero } 0 \rightarrow \text{true}$$

$$\text{E-IszeroSucc} \quad \text{iszero } (\text{succ } nv) \rightarrow \text{false}$$

$$\frac{\text{E-Succ} \quad t \rightarrow t'}{\text{succ } t \rightarrow \text{succ } t'}$$

$$\frac{\text{E-Pred} \quad t \rightarrow t'}{\text{pred } t \rightarrow \text{pred } t'}$$

$$\text{E-PredZero} \quad \text{pred } 0 \rightarrow 0$$

$$\text{E-PredSucc} \quad \text{pred } (\text{succ } nv) \rightarrow nv$$

$$\text{E-IfTrue} \quad \text{if true then } t_2 \text{ else } t_3 \rightarrow t_2$$

$$\text{E-IfFalse} \quad \text{if false then } t_2 \text{ else } t_3 \rightarrow t_3$$

$$\frac{\text{E-If} \quad t_1 \rightarrow t'_1}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3}$$

About small-step semantics

- Note how rules enforce an evaluation ordering
- Two kinds of rules:
 - congruence rules
 - computation rules
- Again, the small-step evaluation relation \rightarrow is defined as the smallest relation that satisfies the set of inference rules we gave.
- If the pair (t, u) is in this relation we say that the **evaluation judgment** $t \rightarrow u$ can be **derived**
- Exercise 1: Draw the derivation tree for the judgment:
`if (iszero (succ 0)) then false else (iszero (succ 0)) \rightarrow
if false then false else (iszero (succ 0))`
- Exercise 2: Continue the steps of one-step evaluation of the above term, until no rule applies

Normal forms

- With small-step semantics, we can define **normal forms** naturally:

Definition (Normal form)

A term t is in a **normal form** if there is no term t' such that $t \rightarrow t'$.

- Normal forms are the meanings of programs
 - if $t \twoheadrightarrow t'$ and t' is a normal form, then t' is the meaning of t
- Some properties of \mathcal{NB} regarding normal forms:
 - every value is a normal form. Proof?

Normal forms

- With small-step semantics, we can define **normal forms** naturally:

Definition (Normal form)

A term t is in a **normal form** if there is no term t' such that $t \rightarrow t'$.

- Normal forms are the meanings of programs
 - if $t \twoheadrightarrow t'$ and t' is a normal form, then t' is the meaning of t
- Some properties of \mathcal{NB} regarding normal forms:
 - every value is a normal form. Proof? Trivial, no evaluation rules that have a value on the left side of \rightarrow

Normal forms

- With small-step semantics, we can define **normal forms** naturally:

Definition (Normal form)

A term t is in a **normal form** if there is no term t' such that $t \rightarrow t'$.

- Normal forms are the meanings of programs
 - if $t \twoheadrightarrow t'$ and t' is a normal form, then t' is the meaning of t
- Some properties of \mathcal{NB} regarding normal forms:
 - every value is a normal form. Proof? Trivial, no evaluation rules that have a value on the left side of \rightarrow
 - does the opposite hold?

Normal forms

- With small-step semantics, we can define **normal forms** naturally:

Definition (Normal form)

A term t is in a **normal form** if there is no term t' such that $t \rightarrow t'$.

- Normal forms are the meanings of programs
 - if $t \twoheadrightarrow t'$ and t' is a normal form, then t' is the meaning of t
- Some properties of \mathcal{NB} regarding normal forms:
 - every value is a normal form. Proof? Trivial, no evaluation rules that have a value on the left side of \rightarrow
 - does the opposite hold? Nope, e.g., `iszero true`

Normal forms

- With small-step semantics, we can define **normal forms** naturally:

Definition (Normal form)

A term t is in a **normal form** if there is no term t' such that $t \rightarrow t'$.

- Normal forms are the meanings of programs
 - if $t \twoheadrightarrow t'$ and t' is a normal form, then t' is the meaning of t
- Some properties of \mathcal{NB} regarding normal forms:
 - every value is a normal form. Proof? Trivial, no evaluation rules that have a value on the left side of \rightarrow
 - does the opposite hold? Nope, e.g., `iszero true`
 - normal forms of a term are unique: u_1, u_2 normal forms,
 $t \twoheadrightarrow u_1 \wedge t \twoheadrightarrow u_2 \implies u_1 = u_2$

Normal forms

- With small-step semantics, we can define **normal forms** naturally:

Definition (Normal form)

A term t is in a **normal form** if there is no term t' such that $t \rightarrow t'$.

- Normal forms are the meanings of programs
 - if $t \twoheadrightarrow t'$ and t' is a normal form, then t' is the meaning of t
- Some properties of \mathcal{NB} regarding normal forms:
 - every value is a normal form. Proof? Trivial, no evaluation rules that have a value on the left side of \rightarrow
 - does the opposite hold? Nope, e.g., `iszero true`
 - normal forms of a term are unique: u_1, u_2 normal forms, $t \twoheadrightarrow u_1 \wedge t \twoheadrightarrow u_2 \implies u_1 = u_2$ (follows if \rightarrow deterministic)

Outline

- 1 Small-step semantics
- 2 Properties of \mathcal{NB}
- 3 Termination

Properties of \mathcal{NB}

Theorem (Determinacy of one-step evaluation)

If $t \rightarrow t_1$ and $t \rightarrow t_2$, then $t_1 = t_2$.

- To prove this, we could manage with induction on the structure of t , but let's look at **induction on derivations**

Induction on proofs/derivation trees

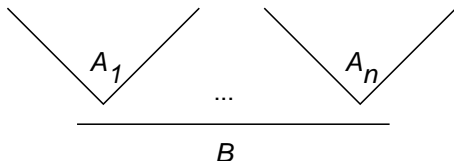
- We defined terms inductively (either directly, or with inference rules)
- Evaluation relation is also defined inductively (using inference rules)
- Just as induction works on the structure of terms, it works on the structure on derivations!
- Derivation tree is a proof, the **witness** or **justification**, of its conclusion. If something holds for all witnesses of a conclusion, it will hold for the conclusion too.
- Thus, a proof is a mathematical object we can manipulate
 - Sequence of formulas, each formula either an axiom or following from previous formulas by a single inference rule
 - Another view: a tree with internal nodes and leaves labeled with formulas (axioms in the leaves)

Proof as a tree

- Assume the inference rule

$$\frac{A_1 \quad A_2 \quad \dots \quad A_n}{B}$$

- A proof of B would then be a tree whose root is B , and proofs of each of A_i are subtrees of the root.



Induction on proofs

- We want to prove that some property $P(\pi)$ is true for all proofs π of some judgment
- To do this, for each derivation rule:

$$\frac{A_1 \quad A_2 \quad \dots \quad A_n}{B}$$

we can assume that for all proofs π_i of A_i , $P(\pi_i)$ holds, and must show $P(\pi)$ for any proof π that extends the proofs π_1, \dots, π_n with one use of the inference rule.

- Thus, for axioms, obviously no assumptions can be used.

Induction principle for \rightarrow in \mathcal{NB}

To prove some property P for all derivations $t \rightarrow t'$, we must show:

- $P(\text{iszero } 0 \rightarrow \text{true})$
- $P(\text{iszero } (\text{succ } nv) \rightarrow \text{false})$
- $P(\text{pred } 0 \rightarrow 0)$
- $P(\text{pred } (\text{succ } nv) \rightarrow nv)$
- $P(\text{if true then } t_2 \text{ else } t_3 \rightarrow t_2)$
- $P(\text{if false then } t_2 \text{ else } t_3 \rightarrow t_3)$
- $P(t \rightarrow t') \implies P(\text{iszero } t \rightarrow \text{iszero } t')$
- $P(t \rightarrow t') \implies P(\text{succ } t \rightarrow \text{succ } t')$
- $P(t \rightarrow t') \implies P(\text{pred } t \rightarrow \text{pred } t')$
- $P(t_1 \rightarrow t'_1) \implies P(\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3)$

Example

Theorem (Determinacy of one-step evaluation)

If $t \rightarrow t'$ and $t \rightarrow t''$, then $t' = t''$.

Proof By induction on derivation of $t \rightarrow t'$.

Case E-IszeroZero: Assume the last (and only in this case) rule used to prove $t \rightarrow t'$ is E-IszeroZero (`iszero 0 \rightarrow true`). Then t is of the form `iszero t_1` , where $t_1 = 0$. The only potentially matching rules that could be used to reduce t to some other term t'' are E-IszeroSucc and E-Iszero. The first could not match as $t_1 = 0$ and thus $t_1 \neq \text{succ } nv$. The second could not match as that rule requires that $t_1 \rightarrow t'_1$ for some t'_1 , but $t_1 = 0$ and already in normal form. Thus, the only rule that can apply is E-IszeroZero, giving $t' = t''$.

All other base cases: Exercise.

example continues

Case E-If: Assume the last rule used to prove $t \rightarrow t'$ is E-If. Then t is of the form $\text{if } t_1 \text{ then } t_2 \text{ else } t_3$ and for some t'_1 , $t_1 \rightarrow t'_1$. Consider now the derivation $t \rightarrow t''$. The last rule cannot be E-IfTrue or E-IfFalse, because in those rules, t_1 is in normal form. The only possible rule that can match is then E-If (the same rule), which gives that $t_1 \rightarrow t''_1$, for some t''_1 . Now $t_1 \rightarrow t'_1$ is a subderivation of $t \rightarrow t'$, and thus by induction hypothesis and $t_1 \rightarrow t''_1$, it follows that $t'_1 = t''_1$. From this, $t' = \text{if } t'_1 \text{ then } t_2 \text{ else } t_3 = \text{if } t''_1 \text{ then } t_2 \text{ else } t_3 = t''$ follows. \square

Outline

- 1 Small-step semantics
- 2 Properties of \mathcal{NB}
- 3 Termination

More properties of \mathcal{NB}

- Show that \rightarrow and \Downarrow define the same relation
 - You can do this at home
- Termination of \rightarrow
 - This we do now

Termination

Theorem (Termination of evaluation)

For all terms t in \mathcal{NB} , $t \rightarrow t'$ for some normal form t' .

- We first need to define a **measure** of terms/machine states.
- Define the **size** of the term as

$$\text{size}(\text{true}) = 1 \qquad \text{size}(\text{false}) = 1 \qquad \text{size}(0) = 1$$

$$\frac{\text{size}(t) = n}{\text{size}(\text{succ } t) = n + 1} \qquad \frac{\text{size}(t) = n}{\text{size}(\text{pred } t) = n + 1}$$

$$\frac{\text{size}(t) = n}{\text{size}(\text{iszero } t) = n + 1}$$

$$\frac{\text{size}(t_1) = n_2 \quad \text{size}(t_2) = n_2 \quad \text{size}(t_3) = n_2}{\text{size}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3) = n_1 + n_2 + n_3 + 1}$$

Proof continues

Lemma

If $t \rightarrow t'$ then $\text{size}(t') < \text{size}(t)$.

Proof.

Induction on the derivation of $t \rightarrow t'$. Easy observation for derivations consisting of a single axiom.

Case IsZero: t must be of the form `iszero` t_1 for some t_1 , and $t_1 \rightarrow t'_1$ for some t'_1 . By the definition of `size`, $\text{size}(t) = \text{size}(t_1) + 1$, and $\text{size}(t') = \text{size}(t'_1) + 1$. By induction hypothesis, $\text{size}(t'_1) < \text{size}(t_1)$, thus $\text{size}(t') < \text{size}(t)$.

Other cases similarly. □

Proof continues

- Now the main proof is easy.

Termination.

The *size* function maps terms to natural numbers. Natural numbers are a *well-founded set*, and thus have no *infinite descending chains*. As $t \rightarrow t' \implies \text{size}(t') < \text{size}(t)$, the existence of an infinite chain of reductions

$$t \rightarrow t' \rightarrow t'' \rightarrow \dots$$

would imply the existence of the infinite decreasing sequence

$$\text{size}(t) \rightarrow \text{size}(t') \rightarrow \text{size}(t'') \rightarrow \dots$$

of natural numbers; a contradiction. □

- This proof structure is typical for termination proofs.