Evaluation Semantics

Robert Harper
Carnegie Mellon University
Spring, 1999

Transition Semantics

- We defined the dynamic semantics of MinML by a transition system $e \mapsto e'$.
  - Specifies computation step-by-step.
  - Facilitates a clean, crisp formulation of type safety (preservation + progress).
- However, the specification is rather low-level.
  - Details of "search" and "execution".

Evaluation Semantics

- Idea: define the relation "e evaluates to v" directly by a set of rules.
  - gloss over the step-by-step details in favor of a more high-level description.
- Evaluation relation: $e \Downarrow v$.
  - Defined to be the least set closed under the following rules.
  - Supports induction on evaluation!

Evaluation Semantics

- Values evaluate to themselves:
  - $n \Downarrow n$
    true $\Downarrow$ true, false $\Downarrow$ false
    fun f(x:τ1):τ2 is e end $\Downarrow$ fun f(x:τ1):τ2 is e end
- Primitive operations:
  - $o(e1,\ldots, en) \Downarrow v$,
    if $e1 \Downarrow v1, \ldots, en \Downarrow vn$, and
    v is the result of applying o to v1, ..., vn.

Evaluation Semantics

- Conditionals
  - if e then e1 else e2 fi $\Downarrow$ v,
    if e $\Downarrow$ true and e1 $\Downarrow$ v.
  - if e then e1 else e2 fi $\Downarrow$ v,
    if e $\Downarrow$ false and e2 $\Downarrow$ v.
- Applications
  - apply(e1, e2) $\Downarrow$ v,
    if e1 $\Downarrow$ v1, e2 $\Downarrow$ v2, and [v1, v2 /f,x]e $\Downarrow$ v,
    where v1 = fun f(x:τ1):τ2 is e end.

Evaluation Semantics

- That's it! So why bother with transition systems?
  - Evaluation semantics is much easier to read and understand.
  - But transition semantics is much easier for language theory, e.g. type safety.

Type Safety in Evaluation Semantics

- We can easily state and prove a version of Preservation
  - if e : τ and e $\Downarrow$ v, then v : τ.
  - proof is by induction on the evaluation relation

Preservation for Evaluation

- e.g., spse e = apply(e1, e2) $\Downarrow$ v.
  - By the defn of evaluation, e1 $\Downarrow$ v1, e2 $\Downarrow$ v2, and [v1,v2/f,x]e $\Downarrow$ v, where v1 = fun f(x:τ1):τ2 is e end.
  - By inversion e1 : τ2-> τ, e2 : τ2 for some τ2.
  - By induction v1 : τ2 -> τ and v2 : τ2.
  - By substitution [v1,v2/f,x]e : τ.
  - By induction, v : τ.

Progress?

- But what is the analogue of Progress for the evaluation semantics?
  - Canonical forms lemma is exactly as before.
  - Want something like this: if e:τ, then either e is a value, or ????
    - Cannot say "there exists v s t e$\Downarrow$v" because not all expressions terminate!

"Wrong" Rules

- One approach: instrument the semantics with rules that check for type errors.
  - e.g., apply(e1,e2) $\Downarrow$ wrong, if e1 $\Downarrow$ v1 and v1 is not a function
  - e.g., if e then e1 else e2 fi $\Downarrow$ wrong, if e$\Downarrow$v and v is not true or false

"Wrong" Rules

- Define typing of values so that wrong is always ill-typed.
  - Contrast treatment of error earlier!
- Preservation implies that if e:τ and e $\Downarrow$ v, then v is not wrong.
  - That is, run-time type errors cannot occur.
- But this is rather roundabout!
  - Must include rules for cases that cannot arise!

Relating Evaluation to Calculation

- Theorem: if e $\Downarrow$ v, then e $\mapsto$* v.
- Proof: it suffices to show that the relation $\mapsto$* is closed under the rules defining the relation $\Downarrow$. That is, we proceed by induction on the definition of evaluation.

Relating Evaluation to Calculation

- e.g., we must show that
  if e1 $\mapsto^*$ v1, e2 $\mapsto^*$ v2, and [v1,v2/f,x]e $\mapsto^*$ v,
  then apply(e1, e2) $\mapsto^*$ v, where v1 = fun f(x:τ1):τ2 is e end.
- apply(e1, e2) $\mapsto^*$ apply(v1, e2) $\mapsto^*$
  apply(v1, v2) $\mapsto^*$ [v1,v2/f,x]e $\mapsto^*$ v

Relating Calculation to Evaluation

- Thm: if e $\mapsto$ e' and e' $\Downarrow$ v, then e $\Downarrow$ v.
- Corollary: if e $\mapsto^*$ v, then e $\Downarrow$ v.
- Proof of thm: by induction on the definition of the transition relation.
- Proof of corollary: by induction on the number of steps from e to v.
- (See notes for details.)

Summary

- Evaluation semantics is more useful for "users".
  - e.g., in a reference manual.
- Transition semantics is more useful for "language designers".
  - e.g., in proofs of type safety.
- The two assign the same values to expressions.