

Chapter 8: Typed Arithmetic Expressions

Types
The Typing Relation
Safety = Progress + Preservation



Reall: Syntax



```
true constant true false if t then t else t conditional constant or zero succ t successor pred t zero test iszero t
```



Evaluation Results



• Values

• Get stuck (i.e., pred false)

values:
true value
false value
numeric value
numeric values:
zero value
successor value



Types of Terms



 Can we tell, without actually evaluating a term, that the term evaluation will not get stuck?



- Distinguish two types of terms:
 - Nat: terms whose results will be a numeric value
 - Bool: terms whose results will be a Boolean value
- "a term t has type T" means that t
 "obviously" (statically) evaluates to a value of T
 - if true then false else true has type Bool
 - pred (succ (pred (succ 0))) has type Nat



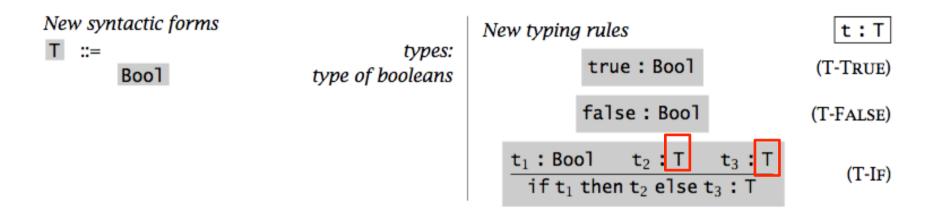


The Typing Relation: t: T



Typing Rule for Booleans







Typing Rules for Numbers



New syntactic forms

T ::= ... types:

Nat type of natural numbers

New typing rules t:T

0: Nat (T-ZERO)

 $\frac{\mathsf{t}_1 : \mathsf{Nat}}{\mathsf{succ}\; \mathsf{t}_1 : \mathsf{Nat}}$

(T-Succ)

 $\frac{\mathsf{t}_1 : \mathsf{Nat}}{\mathsf{pred}\; \mathsf{t}_1 : \mathsf{Nat}}$

(T-PRED)

 $\frac{\mathtt{t}_1 : \mathtt{Nat}}{\mathtt{iszero} \ \mathtt{t}_1 : \mathtt{Bool}}$

(T-IsZero)



Typing Relation: Formal Definition



- **Definition**: the typing relation for arithmetic expressions is the smallest binary relation between terms and types satisfying all instances of the typing rules.
- A term t is typable (or well typed) if there is some T such that t: T.



Inversion Lemma (Generation Lemma)



- Given a valid typing statement, it shows
 - how a proof of this statement could have been generated;
 - a recursive algorithm for calculating the types of terms.

```
    LEMMA [INVERSION OF THE TYPING RELATION]:
    If true: R, then R = Bool.
    If false: R, then R = Bool.
    If if t<sub>1</sub> then t<sub>2</sub> else t<sub>3</sub>: R, then t<sub>1</sub>: Bool, t<sub>2</sub>: R, and t<sub>3</sub>: R.
    If 0: R, then R = Nat.
    If succ t<sub>1</sub>: R, then R = Nat and t<sub>1</sub>: Nat.
    If pred t<sub>1</sub>: R, then R = Nat and t<sub>1</sub>: Nat.
    If iszero t<sub>1</sub>: R, then R = Bool and t<sub>1</sub>: Nat.
```



Typing Derivation



Statements are formal assertions about the typing of programs. Typing rules are implications between statements Derivations are deductions based on typing rules.



Uniqueness of Types



• **Theorem** [Uniqueness of Types]: Each term t has at most one type. That is, if t is typable, then its type is unique.

 Note: later on, we may have a type system where a term may have many types.





Safety = Progress + Preservation



Safety (Soundness)



- By safety, it means well-typed terms do not "go wrong".
- By "go wrong", it means reaching a "stuck state" that is not a final value but where the evaluation rules do not tell what to do next.



Safety + Progress + Preservation



Well-typed terms do not get stuck



- Progress: A well-typed term is not stuck (either it is a value or it can take a step according to the evaluation rules).
- Preservation: If a well-typed term takes a step of evaluation, then the resulting term is also well typed.



Canonical Form



- Lemma [Canonical Forms]:
 - If v is a value of type Bool, then v is either true or false.
 - If v is a value of type Nat, then v is a numeric value according to the grammar for nv.

```
        values:

        true
        true value

        false
        false value

        nv
        numeric value

        nv ::=
        numeric values:

        o
        zero value

        succ nv
        successor value
```



Progress



• Theorem [Progress]: Suppose t is a well-typed term (that is, t : T for some T). Then either t is a value or else there is some t' with $t \rightarrow t'$.

Proof: By induction on a derivation of t : T.



Preservation



• Theorem [Preservation]:

```
If t : T and t \to t', then t' : T.
```

Proof: By induction on a derivation of t: T.

Note: The preservation theorem is often called subject reduction (or subject evaluation)



Homework

国立情報学研究所 Noticed Institute of Informatics

- Read Chapter 8.
- Do Exercises 8.3.7

