CS206 Lecture 14

Substitutions, Matching, Unification

G. Sivakumar
Computer Science Department
IIT Bombay
siva@iitb.ac.in
http://www.cse.iitb.ac.in/~siva

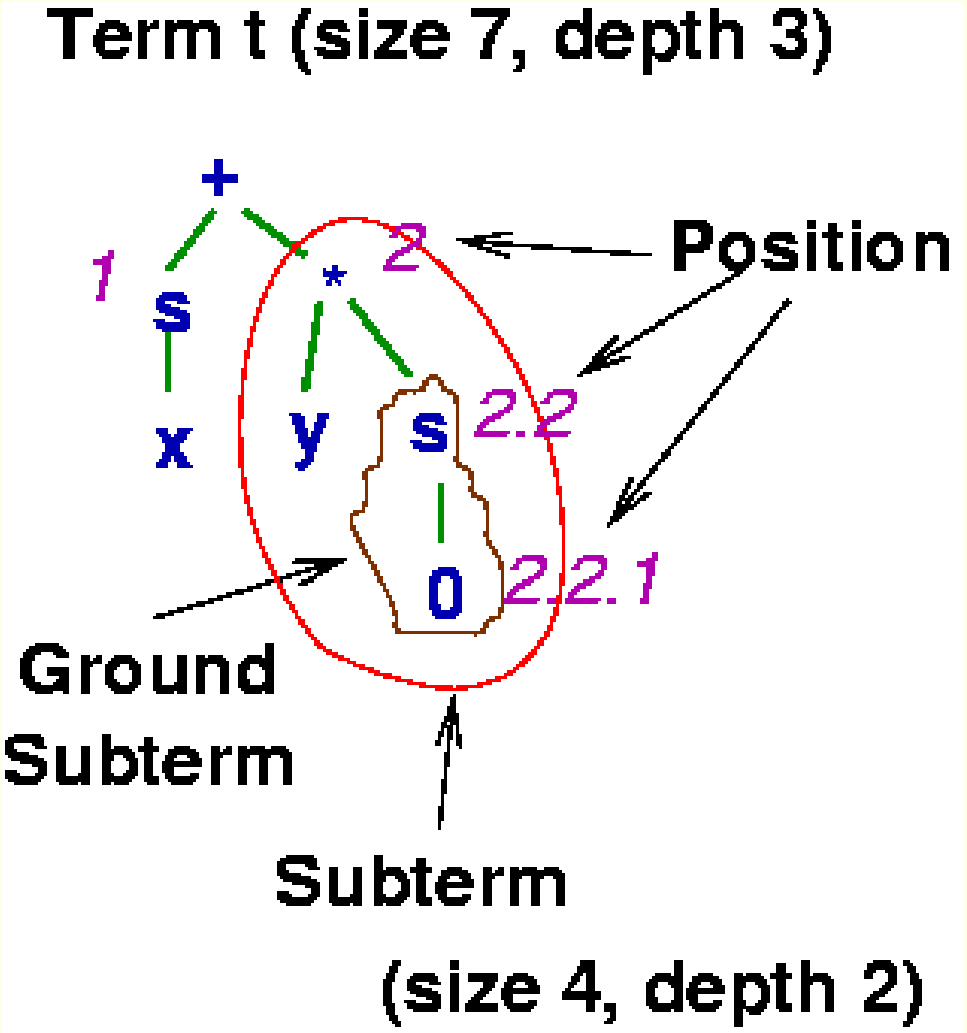
Thu, Jan 30, 2003

Plan for Lecture 13

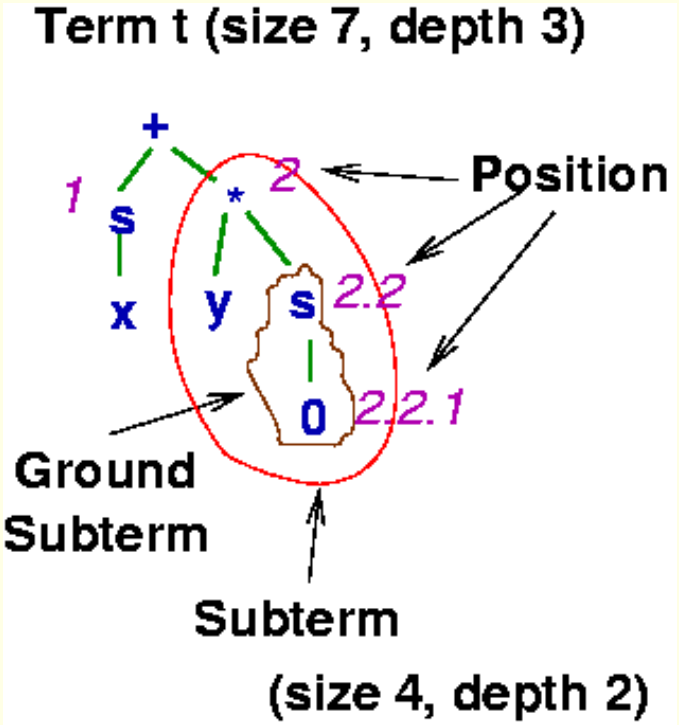
- Substitutions
- Unification

Terms

Consider $t = s(x) + (y * s(0))$



Positions and Subterms



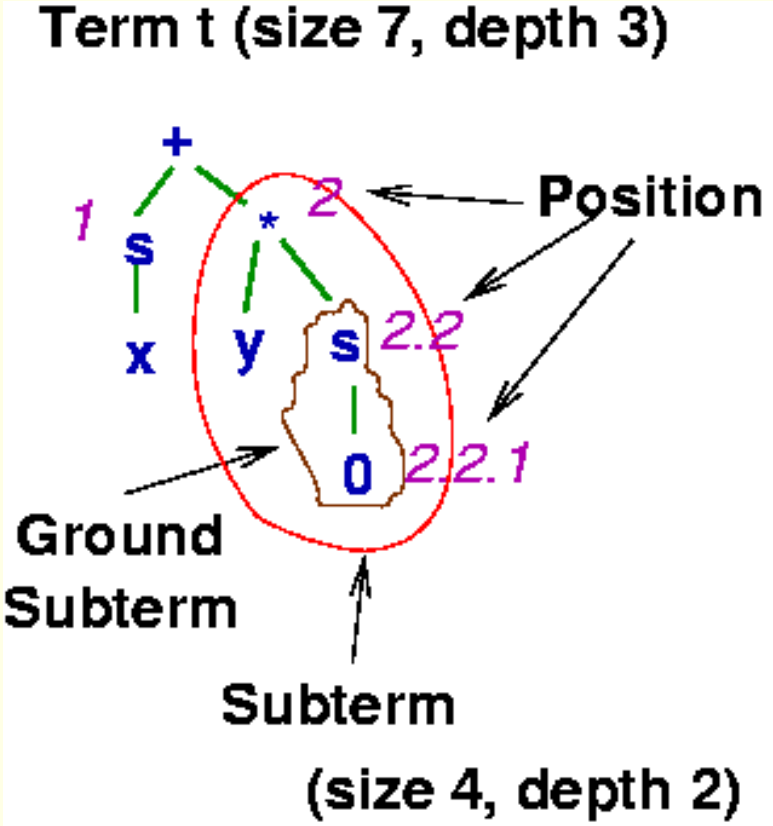
$t/2.2 = s(0)$ is a ground subterm.
$Vars(t/1) = \{x\}$
HomeWork

- Define **positions** in a term.
- Algorithm to extract subterm t/λ at position λ in term t

Replacement

Replacement of a subterm t/λ in a term t at position λ by some other term t_1 is denoted by

$$t[t_1]_\lambda$$



$$t[0]_1 = 0 + (y * s(0))$$

$$t[s(x)]_{2.1} = s(x) + (s(x) * s(0))$$

Substitution

A substitution σ is a mapping from *variables* to **terms**.

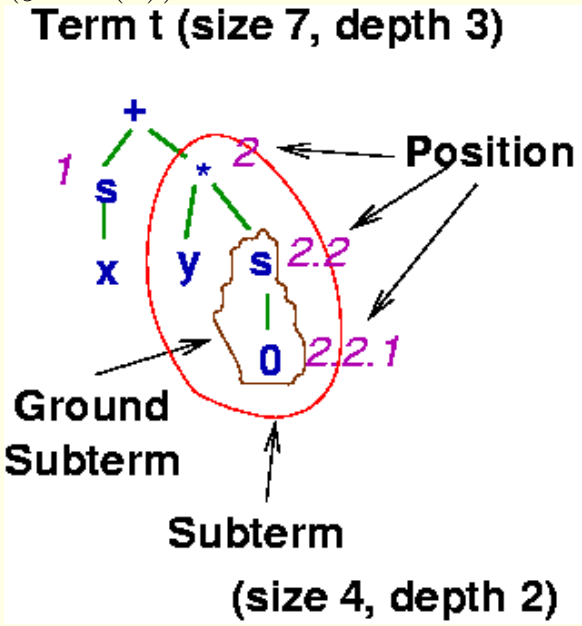
$$\{x \mapsto 0, y \mapsto s(0), z \mapsto v\}$$

Domain of σ is the set of variables $\{x, y, z\}$.
Replacements (Range) of σ is the set of terms $\{0, s(0), v\}$

Applying a Substitution

Replace *variables* by their mapping.

Consider $t = s(x) + (y * s(0))$



and $\sigma = \{x \mapsto 0, y \mapsto s(0), z \mapsto v\}$

Then $t\sigma = s(0) + (s(0) * s(0))$

replace every occurrence of every variable in the substitution by the term with which it is associated.

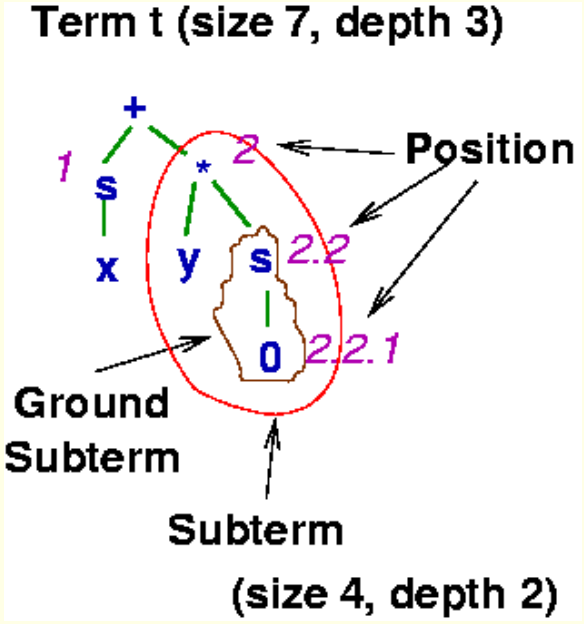
Parallel Replacement

Substiutions are applied in parallel.

Consider

$$\sigma = \{x \mapsto y, y \mapsto s(0), z \mapsto v\}$$

applied to



should give $s(y) + (s(0) * s(0))$

If we replace in **sequence** (replace x first and then replace y) we will get wrong answer $s(0) + (s(0) * s(0))$

Identity and Idempotent Substitution

We do not allow mapping a variable to itself $\{x \mapsto x\}$ explicitly in σ. It is the **default** for variables not in the **domain** of σ.

So, the **identity** substitution σ_{id} is the empty substitution $\sigma_{id} = \{\}$.

For any term $t\sigma_{id} = t$.

A substitution is idempotent (also *pure*) if and only if all replacements are free of the variables in the domain of the substitution. Otherwise, the substitution is impure.

$$\sigma = \{x \mapsto y,\ y \mapsto s(0),\ z \mapsto v\}$$

is impure (non-idempotent).

For **idempotent** substitutions applying in sequence or parallel gives same result.

Also for idempotent σ, we have $(t\sigma)\sigma = t\sigma$ (applying more than once is same as applying once only).

Composing Substitutions

Let

$$\sigma_1 = \{x \mapsto 0,\ y \mapsto s(u),\ z \mapsto v\}$$

and

$$\sigma_2 = \{u \mapsto 0,\ v \mapsto s(0)\}$$

and

$$t = f(x, y, z)$$

Then

$$t\sigma_1 = f(0, s(u), v)$$

$$(t\sigma_1)\sigma_2 = f(0, s(0), s(0))$$

If we define

$$\tau = \sigma_1 \circ \sigma_2 = \{x \mapsto 0,\ y \mapsto s(0),\ z \mapsto s(0),\ u \mapsto 0,\ v \mapsto s(0)\}$$

then $t\tau = f(0, s(0), s(0))$.

Composition of Substitutions

The composition of substitution σ and τ is the substitution (written $\sigma \circ \tau$ or simply $\sigma\tau$) obtained by

- Applying τ to the replacements in σ
- Adjoining to σ the pairs from τ with different variables.
- Deleting any assignments of variable to itself.

$$\sigma = \{x \mapsto 0,\ y \mapsto s(u),\ z \mapsto v,\ x_1 \mapsto y_1\}$$

$$\tau = \{u \mapsto 0,\ v \mapsto s(0),\ z \mapsto 0,\ y_1 \mapsto x_1\}$$

$$\sigma\tau = \{x \mapsto 0,\ y \mapsto s(0),\ z \mapsto s(0),\ u \mapsto 0,\ v \mapsto s(0)\}$$

Does composition preserve idempotence?

Composability

Definition: σ and τ are composable if and only if the variables in the domain of σ are no in the replacements of τ.
The following are not composable.

$$\sigma = \{x \mapsto 0,\ y \mapsto s(0)\}$$

with

$$\tau = \{u \mapsto x,\ v \mapsto s(0)\}$$

Homework

- The composition of composable idempotent substitutions is idempotent.
- Composition is associative.
- Composition has left and right identities.
- Composition is not commutative.

Matching

A (pattern) term t matches a (target) term s iff there is a subsitution σ such that $t\sigma = s$.

Example

$$t = s(x) * y$$

matches

$$s = s(s(0)) * 0$$

using

$$\sigma = \{x \mapsto s(0),\ y \mapsto 0\}$$

Used for **applying rules** to compute **normal forms** in term rewriting systems.

Note: s can have variables. But, the **asymmetric** definition allows replacements only in the pattern term.

Unification

A term t unifies with another term s iff there is a subsitution σ such that $t\sigma = s\sigma$.

Note: Definition is **symmetric**. Can change variables in both terms.

Examples

Term 1	Term 2	Unifier
0	X	$\{X \mapsto 0\}$
p(a, X)	p(Y, b)	$\{X \mapsto b,\ Y \mapsto a\}$
p(f(X), g(Z))	p(f(a), Y)	$\{X \mapsto a,\ Y \mapsto g(Z)\}$
p(f(X), g(Z))	p(f(a), Y)	$\{X \mapsto a,\ Y \mapsto g(b),\ Z \mapsto b\}$

Informal development today of algorithms for matching and unification.