Distributed Unification

Seif Haridi & Peter Van Roy
SICS & UCL

1

Overview

- Centralized unification
  - Rational tree unification
- Distributed unification
  - Local and remote operations
- Proving things
  - Configurations and executions
  - Mapping from distributed to centralized execution
  - How to show the distributed semantics are right
- Safety theorem
- Liveness theorem
- Correctness of distributed unification

2

Notation for terms and actions

- Terms u, v can be variables or nonvariables
  - Variables are denoted by x, y
  - Nonvariables are denoted by s, t
- An action α is a conjunction of primitive actions
- Primitive actions are of four kinds:

$u = v$	Equation	(centralized and distributed)
$u \rho v$	Binding request	(distributed only)
$u \approx v$	Pending equation	
$u \Leftarrow v$	Binding in transit	

3

Notation for the store

- Centralized store:

$$\sigma = \bigcup_i x_i \leftarrow u_i \qquad \text{(where all } x_i \text{ are distinct)}$$

- Distributed store (S is set of sites):

$$\Sigma = \bigcup_{k \in S} \Sigma_k \cup \Gamma$$

$$\Sigma_k = \bigcup_i (x_i \leftarrow u_i)_k \qquad \text{(on site k, where all } x_i \text{ are distinct)}$$

$$\Gamma = \bigcup_{x_i \in lhs(\Sigma)} bound(x_i) \cup \bigcup_{x_i \notin lhs(\Sigma)} unbound(x_i)$$

4

Rational tree unification

- Requires keeping track of variable-variable equations in a memo table μ
- Two operations: adding to the table and checking whether an equation is in the table:
  - ADD(x=y,μ) = $\mu \cup \{x=y\}$
  - ADD(x=t,μ) = μ
  - MEM(x=u,μ) = true if $x=u \in \mu$ or $u=x \in \mu$
  - MEM(x=u,μ) = false otherwise

5

Centralized Semantics

- The centralized execution has action α, store σ, and memo table μ

Structure $\dfrac{\alpha_1 \wedge \alpha_2 \mid \sigma;\mu}{\alpha'_1 \wedge \alpha_2 \mid \sigma';\mu'}$ if $\dfrac{\alpha_1 \mid \sigma;\mu}{\alpha'_1 \mid \sigma';\mu'}$

Congruence
$$\begin{cases} \alpha_1 \wedge \alpha_2 \equiv \alpha_2 \wedge \alpha_1 \\ \text{false} \wedge \alpha \equiv \text{false} \\ \text{true} \wedge \alpha \equiv \alpha \end{cases}$$

Bind $\dfrac{x = u \mid \sigma;\mu}{\text{true} \mid x \leftarrow u, \sigma;\mu}$ $\text{ord}(x) > \text{ord}(u), x \notin \text{lhs}(\sigma)$

- The same structure and congruence rules hold for the distributed action A, store Σ, and memo table M

6

Centralized Semantics

Dereference Test $\dfrac{x = u}{x \leftarrow v, \sigma; \mu} \Bigg| \dfrac{\text{true}}{x \leftarrow v, \sigma; \mu} \text{MEM}(x = u, \mu)$

Dereference Memo $\dfrac{x = u}{x \leftarrow v, \sigma; \mu} \Bigg| \dfrac{v = u}{x \leftarrow v, \sigma; \text{ADD}(x = u, \mu)} \neg \text{MEM}(x = u, \mu)$

Interchange $\dfrac{u = x}{\sigma; \mu} \Bigg| \dfrac{x = u}{\sigma; \mu} \text{ord}(u) < \text{ord}(x), u \notin \text{lhs}(\sigma)$

Identity $\dfrac{x = x}{\sigma; \mu} \Bigg| \dfrac{\text{true}}{\sigma; \mu}$

Incompatibility $\dfrac{s = t}{\sigma; \mu} \Bigg| \dfrac{\text{false}}{\sigma; \mu} \text{incompatible}(s, t)$

Decomposition $\dfrac{s = t}{\sigma; \mu} \Bigg| \dfrac{\bigwedge_i u_i = v_i}{\sigma; \mu} \text{compatible}(s, t)$

7

Distributed Semantics: Local Operations

Dereference Test $\dfrac{(x = u)_k}{(x \leftarrow v)_k, \Sigma; M} \Bigg| \dfrac{\text{true}_k}{(x \leftarrow v)_k, \Sigma; M} \text{MEM}(x = u, M_k)$

Dereference Memo $\dfrac{(x = u)_k}{(x \leftarrow v)_k, \Sigma; M_k \cup M} \Bigg| \dfrac{(v = u)_k}{(x \leftarrow v)_k, \Sigma; \text{ADD}(x = u, M_k) \cup M} \neg \text{MEM}(x = u, M_k)$

Interchange $\dfrac{(u = x)_k}{\Sigma; M} \Bigg| \dfrac{(x = u)_k}{\Sigma; M} \text{ord}(u) < \text{ord}(x), u \notin \text{lhs}(\Sigma_k)$

Identity $\dfrac{(x = x)_k}{\Sigma; M} \Bigg| \dfrac{\text{true}_k}{\Sigma; M}$

Incompatibility $\dfrac{(s = t)_k}{\Sigma; M} \Bigg| \dfrac{\text{false}_k}{\Sigma; M} \text{incompatible}(s, t)$

Decomposition $\dfrac{(s = t)_k}{\Sigma; M} \Bigg| \dfrac{\bigwedge_i (u_i = v_i)_k}{\Sigma; M} \text{compatible}(s, t)$

8

Distributed Semantics: Remote Operations

Initiate $\dfrac{(x = u)_k}{\Sigma; M} \Big| \dfrac{(x\ \mathsf{p}\ u)_k \wedge (x \approx u)_k}{\Sigma; M}\ \mathrm{ord}(x) > \mathrm{ord}(u), x \notin \mathrm{lhs}(\Sigma_k)$

Win $\dfrac{(x\ \mathsf{p}\ u)_k}{\mathrm{unbound}(x), \Sigma; M} \Big| \dfrac{\bigwedge_{k \in S} (x \Leftarrow u)_k}{\mathrm{bound}(x), \Sigma; M}$

Lose $\dfrac{(x\ \mathsf{p}\ u)_k}{\mathrm{bound}(x), \Sigma; M} \Big| \dfrac{\mathrm{true}_k}{\mathrm{bound}(x), \Sigma; M}$

Arrival $\dfrac{(x \Leftarrow u)_k}{\Sigma; M} \Big| \dfrac{\mathrm{true}_k}{(x \leftarrow u)_k, \Sigma; M}\ x \in \mathrm{var}(\Sigma_k)$

Resume $\dfrac{(x \approx u)_k}{(x \leftarrow v)_k, \Sigma; M} \Big| \dfrac{(x = u)_k}{(x \leftarrow v)_k, \Sigma; M}$

9

Configurations and Executions

- A configuration c is a tuple containing an action α, a store σ, and a memo table μ, we write: $c=(\alpha;\sigma;\mu)$
- A reduction of rule R is a pair of configurations such that the first configuration c matches R's firing condition and the second configuration c' is the result after firing R:

$$c \xrightarrow{R} c'$$

- An execution e is a sequence of configurations successively reduced by a given set of rules:

$$c_1 \xrightarrow{R_1} c_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{n-1}} c_n$$

- We speak of centralized executions (using centralized semantics) and distributed executions (using distributed semantics)

10

Mapping from distributed to centralized configurations

- We map primitive actions and bindings from a distributed configuration to a corresponding centralized configuration:

Primitive actions	$(u = v)_k$	$u = v$
	$(x \rho u)_k$	true
	$(x \approx u)_k$	$x = u$
	$(x \Leftarrow u)_k$	$x \leftarrow u$ (in store)
Bindings	$(x \leftarrow u)_k$	$x \leftarrow u$ (in store)

- In this way we map any distributed configuration to a centralized one:

$$(A;\Sigma;M) \xrightarrow{m} (\alpha;\sigma;\mu) = (m_a(A); m_s(A,\Sigma); m_m(M))$$

$$\mu = m_m(M) = \bigcup_{k \in S} M_k$$

11

How to show that the distributed semantics are right

$$\begin{array}{lccc} e: & (A;\Sigma;M) & \xrightarrow{R_1,R_2,\ldots,R_m} & (A';\Sigma';M') \\ & m \downarrow & & m \downarrow \\ m(e): & (\alpha;\sigma;\mu) & \xrightarrow{R'_1,R'_2,\ldots,R'_n} & (\alpha';\sigma';\mu') \end{array}$$

- For any conjunction of equations, we must show that all distributed executions that start with these equations placed on any sites will terminate and give a result equivalent to a centralized execution
  - Safety: Any distributed execution maps to a centralized execution
  - Liveness: Any distributed execution makes progress, that is, the corresponding centralized execution advances
  - Termination: Any distributed execution that starts with a given conjunction of equations will terminate

12

Safety theorem

- Theorem: If e is any distributed execution, then m(e) is a centralized execution and the rule sequence of m(e) is constructed as follows from e

	Distributed rule	Centralized rule
Local rules	Dereference Test	Dereference Test
	Dereference Memo	Dereference Memo
	Interchange	Interchange
	Identity	Identity
	Incompatibility	Incompatibility
	Decomposition	Decomposition
Remote rules	Initiate	Skip
	Win	Bind
	Lose	Skip
	Arrival	Skip
	Resume	Skip

- Local rules and the Win rule are called "progressing" rules since they advance the centralized execution

Decomposition

- If the distributed execution does a decomposition (DEC), then there exists a centralized rule that reduces the mapped initial configuration to the mapped result configuration
- The centralized rule is a decomposition as well

$$\begin{array}{ccc} \dfrac{(s=t)_k \wedge A}{\Sigma;M} & \xrightarrow{DEC} & \dfrac{\bigwedge_i (u_i=v_i)_k \wedge A}{\Sigma;M} \\ \downarrow & & \downarrow \\ \dfrac{(s=t) \wedge m_a(A)}{m_s(A,\Sigma);m_m(M)} & \xrightarrow{?} & \dfrac{\bigwedge_i (u_i=v_i) \wedge m_a(A)}{m_s(A,\Sigma);m_m(M)} \end{array}$$

- All local rules are handled in the same way

Initiate

$$\frac{(x = u)_k \wedge A}{\Sigma; M} \xrightarrow{I} \frac{(x \; p \; u)_k \wedge (x \approx u)_k \wedge A}{\Sigma; M}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\frac{(x = u) \wedge m_a(A)}{m_s(A,\Sigma); m_m(M)} \xrightarrow{?} \frac{(x = u) \wedge m_a(A)}{m_s(A,\Sigma); m_m(M)}$$

- The centralized rule is a skip
- After the centralized skip, we know ord(x)>ord(u), x∉lhs(Σ)

15

Win

$$\frac{(x \; p \; u)_k \wedge A}{\text{unbound}(x), \Sigma; M} \xrightarrow{W} \frac{\bigwedge_{s \in S} (x \Leftarrow u)_s \wedge A}{\text{bound}(x), \Sigma; M}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\frac{m_a(A)}{m_s(A,\Sigma); m_m(M)} \xrightarrow{?} \frac{m_a(A)}{(x \leftarrow u), m_s(A,\Sigma); m_m(M)}$$

- (x=u) is in $m_a(A)$ because an initiate has been done and x is unbound
- ord(x)>ord(u) because the initiate requires this condition
- x∉lhs($m_s(A,\Sigma)$) because the initiate requires this condition and no Win has been reduced yet (unbound(x))

16

Lose

$$\frac{(x\ \mathrm{p}\ u)_k \wedge A}{\mathrm{bound}(x) \wedge \Sigma; M} \xrightarrow{L} \frac{\mathrm{true}_k \wedge A}{\mathrm{bound}(x) \wedge \Sigma; M}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\frac{m_a(A)}{m_s(A,\Sigma); m_m(M)} \xrightarrow{?} \frac{m_a(A)}{m_s(A,\Sigma); m_m(M)}$$

- The centralized rule is a skip

17

Arrival

$$\frac{(x \Leftarrow u)_k \wedge A}{\Sigma; M} \xrightarrow{L} \frac{\mathrm{true}_k \wedge A}{(x \leftarrow u)_k, \Sigma; M}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\frac{m_a(A)}{(x \leftarrow u)_k, m_s(A,\Sigma); m_m(M)} \xrightarrow{?} \frac{m_a(A)}{(x \leftarrow u)_k, m_s(A,\Sigma); m_m(M)}$$

- The centralized rule is a skip

18

Relations between configurations and stores

- For the liveness theorem, we need some notions from equality theory
- Configuration c_1 entails c_2 if:
  - c_1=(false;_;_), or
  - c_1=(_;σ_1;_) and c_2 =(_;σ_2;_) and σ_1 entails σ_2
- Store σ_1 entails σ_2 if $LF(\sigma_1) \rightarrow LF(\sigma_2)$
  - $LF(\emptyset) = true$
  - $LF((x \leftarrow y,\sigma)) = x=y \wedge LF(\sigma)$
- Configuration c_1 is equivalent to c_2 if c_1 entails c_2 and c_2 entails c_1

19

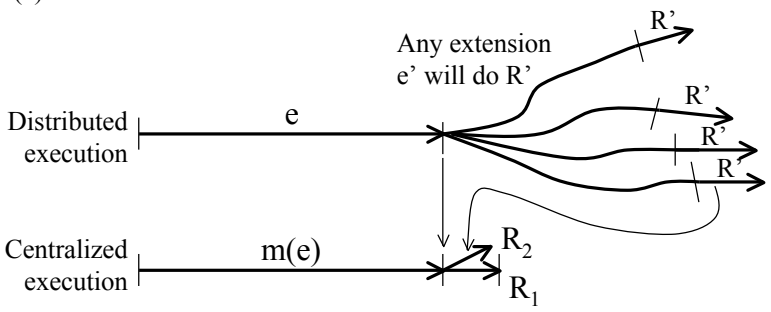
Liveness theorem

- Theorem: Given a centralized execution with start and end configurations c_1 and c_2, then any distributed execution starting with an initial configuration that places c_1 will eventually result in a configuration c_3 such that $m(c_3)$ entails c_2.
- Entailment lemma: Given a centralized execution whose end configuration is c_1, then any centralized execution starting with the same initial configuration will eventually result in a configuration c_2 such that c_2 entails c_1.
- Progress lemma: Given any distributed execution e such that m(e) is nonterminal, then continuing e will always eventually give e' such that m(e') continues m(e).

20

Progress lemma

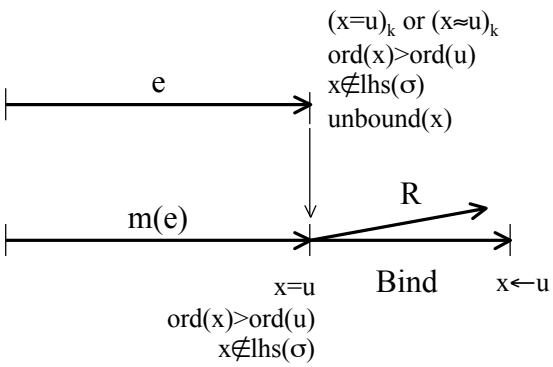
- Given any distributed execution e such that m(e) is nonterminal, then continuing e will always eventually give e' such that m(e') continues m(e)



- Given e, m(e) can do R_1, therefore e will eventually do R', which maps to R_2. Therefore e will make progress!

21

Progress lemma - Bind rule



$(x=u)_k$
Initiate is applicable. After the initiate, Win or Lose is applicable.

$(x\approx u)_k$
Action $(x<u)_k$ also exists. Therefore, Win or Lose is applicable.

If Lose is applicable then a Dereference rule will be applicable and applied.

- If Bind x=u can be applied in the centralized execution, then the distributed execution will apply Bind, Deref Test or Deref Memo

22

Progress lemma - Other rules

- For all other rules in the centralized execution except for Dereference, the corresponding rule will become applicable in the distributed execution
- The dereference rules are either applicable immediately or after the binding arrives (Arrival rule)
- Therefore, “performing an operation” in the centralized execution implies that a corresponding operation will be performed in the distributed execution

23

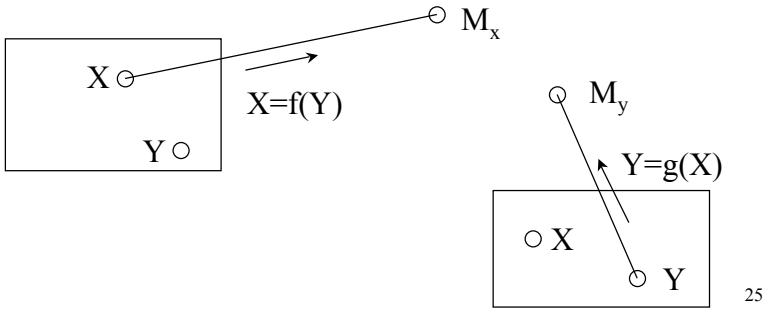
Correctness theorem

- Theorem: For any terminating centralized execution with initial equations α and terminal store σ, all distributed executions with corresponding initial configuration will terminate in a store Σ such that $m(\Sigma) \equiv \sigma$
- Proof: From the progress lemma, eventually the distributed execution maps to a terminating centralized execution:
  - The distributed execution cannot execute any more progressing rules, since this would mean that the centralized execution is not terminating
  - The nonprogressing rules do only finite extra work: bind requests, bindings in transit, and pending equations can all be reduced once
  - Therefore the distributed execution terminates
- Since the centralized execution does unification, we deduce that $m(\Sigma) \equiv \sigma$

24

Distributed occur check

- Occur check must be done at the manager, cannot be done at the proxy
- Manager must lock all variables in the term before doing the binding, and unlock afterwards
- Occur check requires global synchronization, is not a local operation



25

Rules for implementing occur check

- The ¬occur(x,u,σ) condition is not satisfied
- Occur check should be done at the Win rule with θ, not at the Initiate rule with θ_k
- Implementing the Win rule requires global synchronization (expensive!)

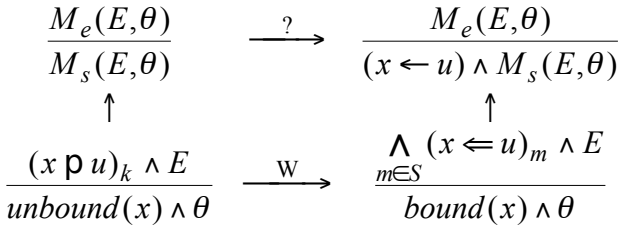
Bind $\dfrac{x = u}{\sigma} \Big| \dfrac{true}{x \leftarrow u, \sigma}$ ord(x) > ord(u), $x \notin lhs(\sigma)$, $\neg occur(x, u, \sigma)$

Initiate $\dfrac{(x = u)_k}{\theta} \Big| \dfrac{(x \; \mathsf{p} \; u)_k \wedge (x \approx u)_k}{\theta}$ ord(x) > ord(u), $x \notin lhs(\theta_k)$, $\neg occur(x, u, \theta_k)$

Win $\dfrac{(x \; \mathsf{p} \; u)_k}{unbound(x), \theta} \Big| \dfrac{\bigwedge_{k \in S} (x \Leftarrow u)_k}{bound(x), \theta}$

26

Win (with occur check)



- (x=u) is in $M_e(E,\theta)$ because an initiate has been done and x is unbound
- ord(x)>ord(u) because the initiate requires this condition
- x∉lhs($M_s(E,\theta)$) since no Win has been reduced yet (unbound(x))

∀ ¬occur(x,u, $M_s(E,\theta)$) is not satisfied

27