Denotational Semantics for a Probability Timed Shared-Variable Language

Huibiao Zhu[1] Jeff Sanders[2] Jifeng He[1] Shengchao Qin[3]

[1]Shanghai Key Laboratory of Trustworthy Computing
Software Engineering Institute, East China Normal University
[2]African Institute for Mathematical Sciences
[3]School of Computing, University of Teesside

UTP 2012, 27-28 August, 2012

Outline

Outline

Background

1. **Complex Software Systems:** (a) Probability; (b) Real-time; (c) Shared-Variable Concurrency

   We have proposed a language PTSC, which integrates these features.

2. **Semantic Study:** We have studied the operational semantics, algebraic semantics and their linking theories for PTSC.

Motivation

(1) PTSC integrates these features:

   (a) Probability;

   (b) Real-time;

   (c) Shared-Variable Concurrency

(2) It is challenging to study the denotational semantics for PTSC.

The Syntax of PTSC

$$P ::= \textbf{Skip} \mid x := e \mid \textbf{if } b \textbf{ then } P \textbf{ else } Q$$
$$\mid \textbf{while } b \textbf{ do } P \mid @b\ P \mid \#n\ P \mid P\ ;\ P$$
$$\mid P \sqcap P \mid P \sqcap_p P \mid P \parallel P \mid P \parallel_p P$$

Five types of guarded choice:

(1) The first type is composed of a set of assignment-guarded components.

$$[\!]_{i \in I}\{[p_i]\ choice_{j \in J_i}(b_{ij} \& (x_{ij} := e_{ij})\ P_{ij})\}$$

Healthiness conditions:

(a) $\forall i \bullet (\bigvee_{j \in J_i} b_{ij} = true)$ and
$(\forall j_1, j_2 \in J_i \bullet (j_1 \neq j_2) \Rightarrow ((b_{ij_1} \wedge b_{ij_2}) = false))$

(b) $\Sigma_{i \in I}\ p_i = 1$

The Syntax of PTSC (2)

Five types of guarded choice:

(2) The second type is composed of a set of event-guarded components.

$$[\!]_{i \in I}\{@b_i\, P_i\}$$

(3) The third type is composed of one time-delay component.

$$[\!]\{\#1\ R\}$$

(4) The fourth type is the guarded choice composition of the first and second type of guarded choice.

$$[\!]_{i \in I}\{[p_i]\ choice_{j \in J_i}(b_{ij} \& (x_{ij} := e_{ij})\, P_{ij})\}$$
$$[\!][\!]_{k \in K}\{@b_k\, Q_k\}$$

The Syntax of PTSC (3)

Five types of guarded choice:

(5) The fifth type is the compound of the second and third type of guarded choice.

$$[\!]_{i\in I}\{@b_i\, P_i\}[\!]\{\#1\ R\}$$

Example: Let

$$\begin{aligned} & P \\ = \ & [\!]\{\ [0.7]choice(\ true\&(x := 5)\, P_1\ )\,, \\ & \quad [0.3]choice(\ (x > 2)\&(x := x)\, P_2,\ \ (x \leq 2)\&(x := x)\, P_3\ ) \\ & \quad \} \end{aligned}$$

Outline

Snapshot

A snapshot is expressed as a triple:

$$(tag,\ p,\ \sigma)$$

where:

(1) σ stands for the contributed state. These states are contributed by the program itself or the environment.

(2) *tag* can be 0, 0^-, 1 and $\surd$.

- (a) Flag 0 indicates that the contribution of σ is due to the environment.
- (b) Flag 1 indicates the contribution of σ is due to the process itself.
- (c) Flag $\surd$ indicates that time advances one unit and the state σ is the same as the previous one.
- (d) Flag 0^- is used to model the case that after the environment actions, the subsequent process will be assignment.

(3) For the second element, it is used to express the probability of the contributed state.

- (a) If $tag = \surd$, it will be $\emptyset$, indicating that we do not need to consider the probability for time delay.
- (b) If $tag = 0, 0^-, 1$, the second element p will be the probability of the contributing the state σ.

The Concept of P^{-}-tree

Definition (P^{-}-tree)

(1) st is P^{-}-tree, where st stands for the execution state. Here, st can be **div**, **wait** or **ter**.

(2) $\{\!|(tag, p_i, \sigma_i) : U_i \mid i \in I \wedge \Sigma_{i \in I}\, p_i = 1|\!\}$ is P^{-}-tree if each element in every U_i is P^{-}-tree.

(3) $\{\!|(\surd, \emptyset, \sigma) : U\}|\!\}$ is P^{-}-tree if each element in U is P^{-}-tree.

Note that $st \in U_i$(or U) **iff** $U_i = \{st\}$ (or $U = \{st\}$).

P^{-}-tree: Example (1)

Example: Let

Q_1
$=_{df}\ x := 1\ ;\ \#1\ ;$
$\quad x := x + 1 \sqcap_{0.4} x := x + 2.$

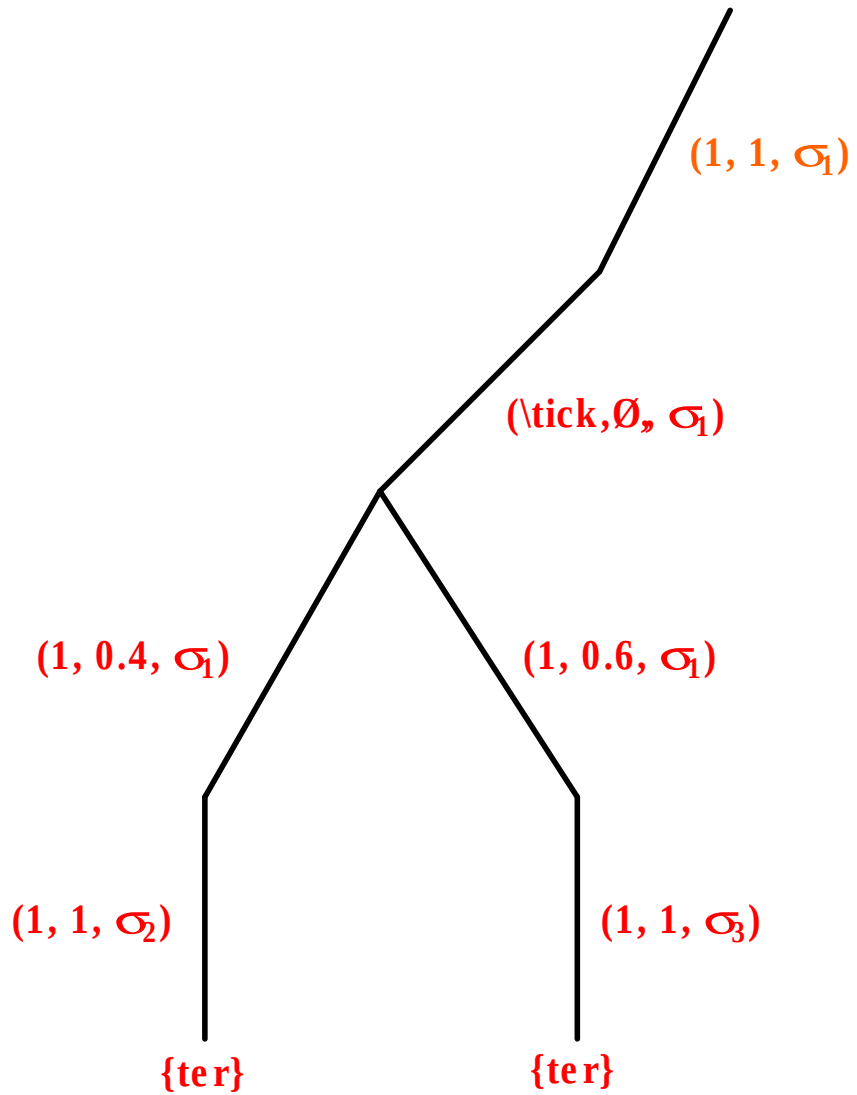
We find that below is one P^{-}-tree of program Q_1.

$$\{\!|(1, 1, \sigma_1) : \{T_0\}|\!\}$$

where,

$$\begin{aligned} T_0 &= \{\!|(\surd, \emptyset, \sigma_1) : \{T_1\}|\!\} \quad \text{and} \\ T_1 &= \{\!|(1, 0.4, \sigma_1) : \{T_{1,1}\}, \\ &\qquad (1, 0.6, \sigma_1) : \{T_{1,2}\}|\!\} \\ T_{1,1} &= \{\!|(1, 1, \sigma_2) : \{ter\}|\!\} \\ T_{1,2} &= \{\!|(1, 1, \sigma_3) : \{ter\}|\!\} \end{aligned}$$

Here, $\sigma_1 = \{x \mapsto 1\}$
$\sigma_2 = \{x \mapsto 2\}$
$\sigma_3 = \{x \mapsto 3\}.$



P^{-}-tree: Example (2)

Example: Let

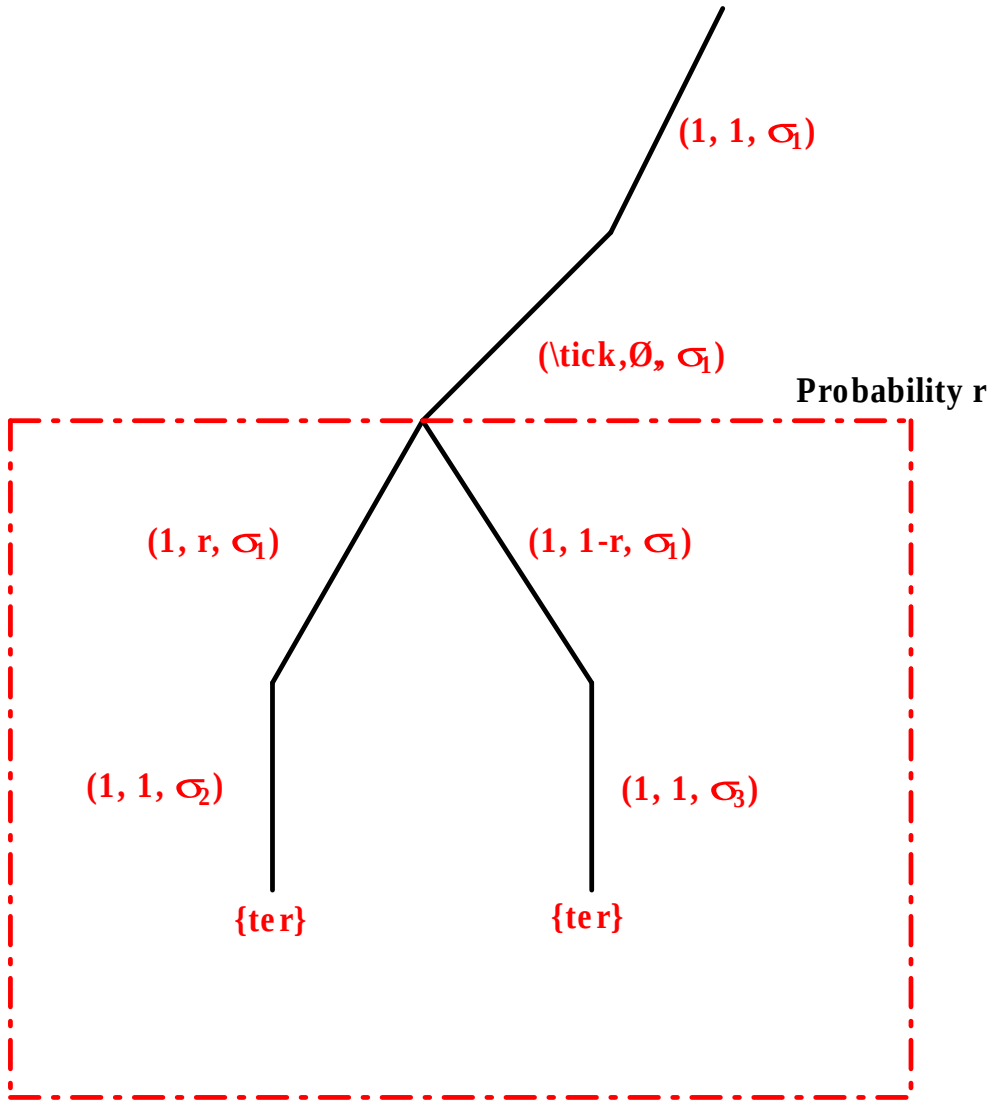
Q_2
$=_{df}$ $x := 1\ ;\ \#1\ ;$
$x := x + 1 \sqcap x := x + 2.$

Below is the P^{-}-tree of program Q_2.

$$\{\!|(1, 1, \sigma_1) : \{T_2\}|\!\}$$

where,

T_2
$= \{\!|(\surd, \emptyset, \sigma_1) : \{T_{3,r} | 0 \leq r \leq 1\}|\!\}$
and
$T_{3,r}$
$= \{\!|(1, r, \sigma_1) : \{T_{1,1}\},$
$(1, 1 - r, \sigma_1) : \{T_{1,2}\}|\!\}$



P^--tree: Example (3)

Example: Let $Q_1 = x := 1 \sqcap_{0.4} x := 2$,

$Q_2 = y := 1 \sqcap_{0.3} y := 2$, $\quad Q = Q_1 \parallel Q_2$.

Consider the P^--trees for process Q_1, Q_2 and Q respectively.

We consider the case that, for process Q, the assignment in Q_1 is scheduled first. Below is one P^--tree for Q_1 at the initial state (tag, μ, σ_0).

$$(tag, \mu, \sigma_0) : \{\ \{\!|\ (1, 0.4, \sigma_0) : \{\ \{\!|(1, 1, \sigma_1) : \{ter\}|\!\}\ \},\ (1, 0.6, \sigma_0) : \{\ \{\!|(1, 1, \sigma_2) : \{ter\}|\!\}\ \}\ |\!\}\ \}$$

where, $\sigma_0 = \{x \mapsto 0,\ y \mapsto 0\}$, $\sigma_1 = \{x \mapsto 1,\ y \mapsto 0\}$, $\sigma_2 = \{x \mapsto 2,\ y \mapsto 0\}$.

As Q_1 is scheduled first, similarly, below is the corresponding P^--tree for Q_2 at the initial state (tag, μ, σ_0).

$$(tag, \mu, \sigma_0) : \{\ \{\!|\ (0^-, 0.4, \sigma_0) : \{\ \{\!|(0^-, 1, \sigma_1) : \{T_2\}|\!\}\ \},\ (0^-, 0.6, \sigma_0) : \{\ \{\!|(0^-, 1, \sigma_1) : \{T_2'\}|\!\}\ \}\ |\!\}\ \}$$

P^--tree: Example (3)

where, $T_2 = \{\!\|\ (1, 0.3, \sigma_1) : \{\{\!|(1, 1, \sigma_1') : \{ter\}|\!\}\},$
$(1, 0.7, \sigma_1) : \{\{\!|(1, 1, \sigma_1'') : \{ter\}|\!\}\}\ \|\!\}$

$T_2' = \{\!\|\ (1, 0.3, \sigma_2) : \{\{\!|(1, 1, \sigma_2') : \{ter\}|\!\}\},$
$(1, 0.7, \sigma_2) : \{\{\!|(1, 1, \sigma_2'') : \{ter\}|\!\}\}\ \|\!\}$

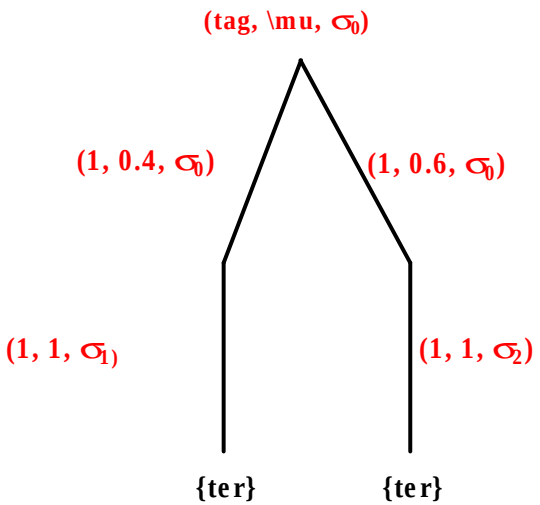
where, $\sigma_1' = \{x \mapsto 1,\ y \mapsto 1\}, \quad \sigma_1'' = \{x \mapsto 1,\ y \mapsto 2\}$

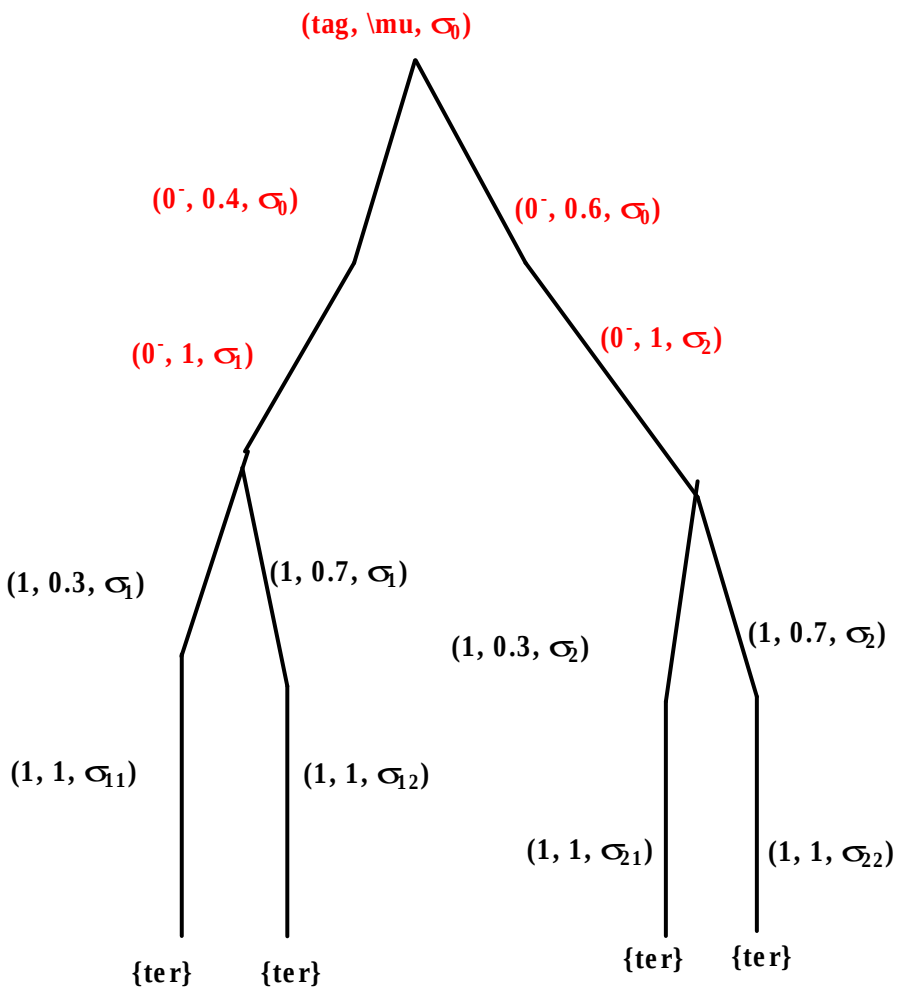
$\sigma_2' = \{x \mapsto 2,\ y \mapsto 1\}, \quad \sigma_2'' = \{x \mapsto 2,\ y \mapsto 2\}$

Hence, below is the corresponding P^--tree for process Q, which is the merge of the P^--tree of process Q_1 and the corresponding P^--tree of Q_2.

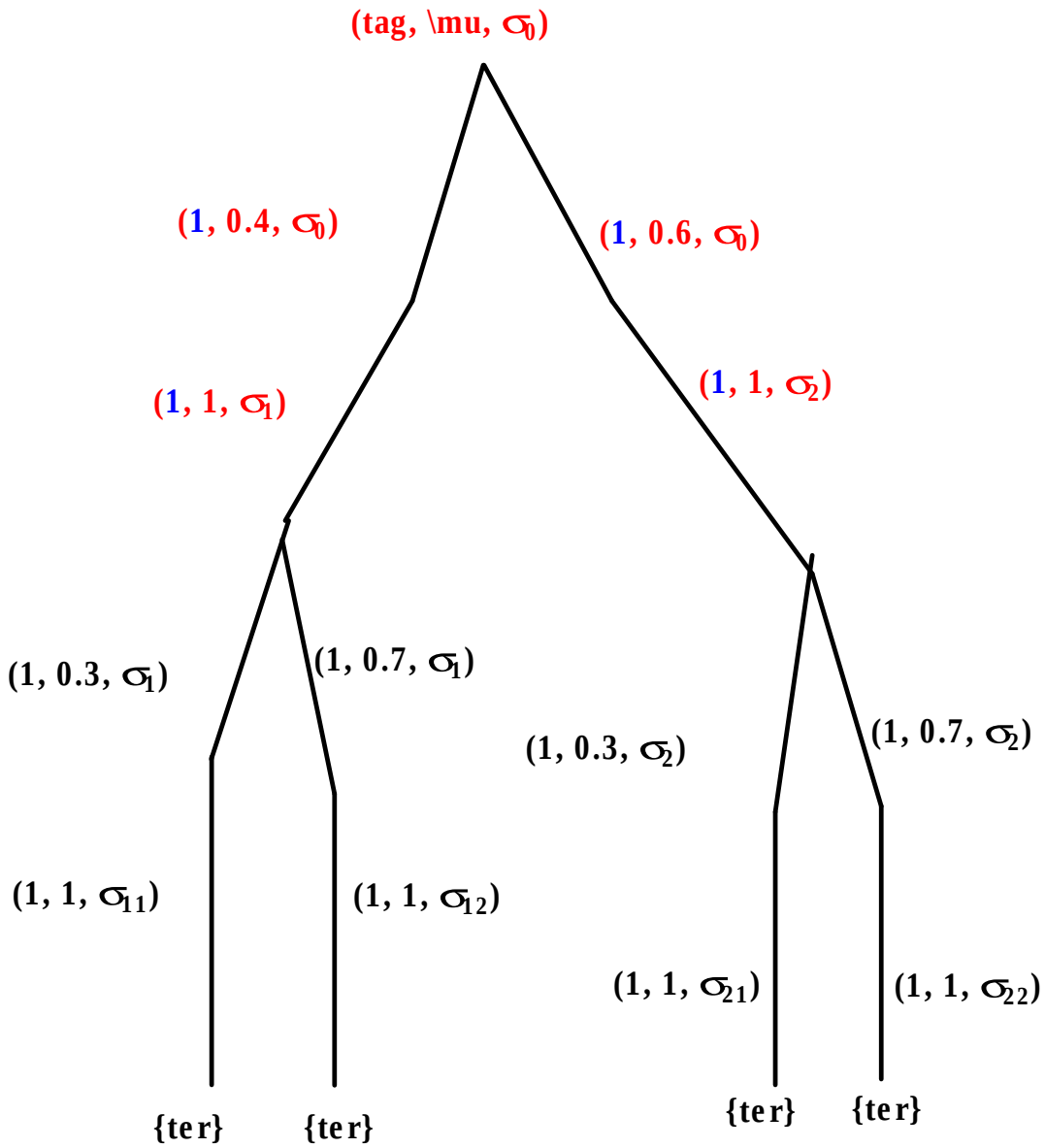
$(tag, \mu, \sigma_0) : \{\ \{\!\|\ (1, 0.4, \sigma_0) : \{\ \{\!|(1, 1, \sigma_1) : \{T_2\}|\!\}\ \},$
$(1, 0.6, \sigma_0) : \{\ \{\!|(1, 1, \sigma_2) : \{T_2'\}|\!\}\ \}$
$\|\!\}\ \}$

P^{-}-tree: Example (3)





P^{-}-tree: Example (3)



Definition (The Concept of P-tree)

$(tag, \mu, \sigma) : U$ is P-tree if each element in U is P^--tree. □

(1) P-tree is composed of an initial snapshot and a set of P^--trees.

(2) The P-tree $(tag, \mu, \sigma) : U$ indicates that each P^--tree in U is initially at the state shown in snapshot (tag, μ, σ).

Example

Example: Let $P =_{df} x := 0 \sqcap x := 1$ and
$Q =_{df} y := 0 \sqcap_{0.5} y := 1.$

Now we want to calculate $Prob(P; Q, x = y)$ and $Prob(Q; P, x = y)$.

(1) First, we consider the P-tree for program $P; Q$.

$$(tag, \mu, \sigma) : \{ \ \{\!| \ (1, r, \sigma) : \{\{\!|(1, 1, \sigma_0) : \{T_1\}|\!\}\}, \\ (1, 1-r, \sigma) : \{\{\!|(1, 1, \sigma_1) : \{T_2\}|\!\}\} \\ |\!\} \ \mid \ 0 \le r \le 1 \ \},$$

$$T_1 = \{\!| \ (1, 0.5, \sigma_0) : \{\{\!|(1, 1, \sigma_0') : \{ter\}|\!\}\}, \ (1, 0.5, \sigma_0) : \{\{\!|(1, 1, \sigma_0'') : \{ter\}|\!\}\} \ |\!\}$$
$$T_2 = \{\!| \ (1, 0.5, \sigma_1) : \{\{\!|(1, 1, \sigma_1') : \{ter\}|\!\}\}, \ (1, 0.5, \sigma_1) : \{\{\!|(1, 1, \sigma_1'') : \{ter\}|\!\}\} \ |\!\}$$

$\sigma = \{x \mapsto -1, \ y \mapsto -1\},$
$\sigma_0 = \{x \mapsto 0, \ y \mapsto -1\}, \quad \sigma_0' = \{x \mapsto 0, \ y \mapsto 0\}, \quad \sigma_0'' = \{x \mapsto 0, \ y \mapsto 1\},$
$\sigma_1 = \{x \mapsto 1, \ y \mapsto -1\}, \quad \sigma_1' = \{x \mapsto 1, \ y \mapsto 0\}, \quad \sigma_1'' = \{x \mapsto 1, \ y \mapsto 1\},$

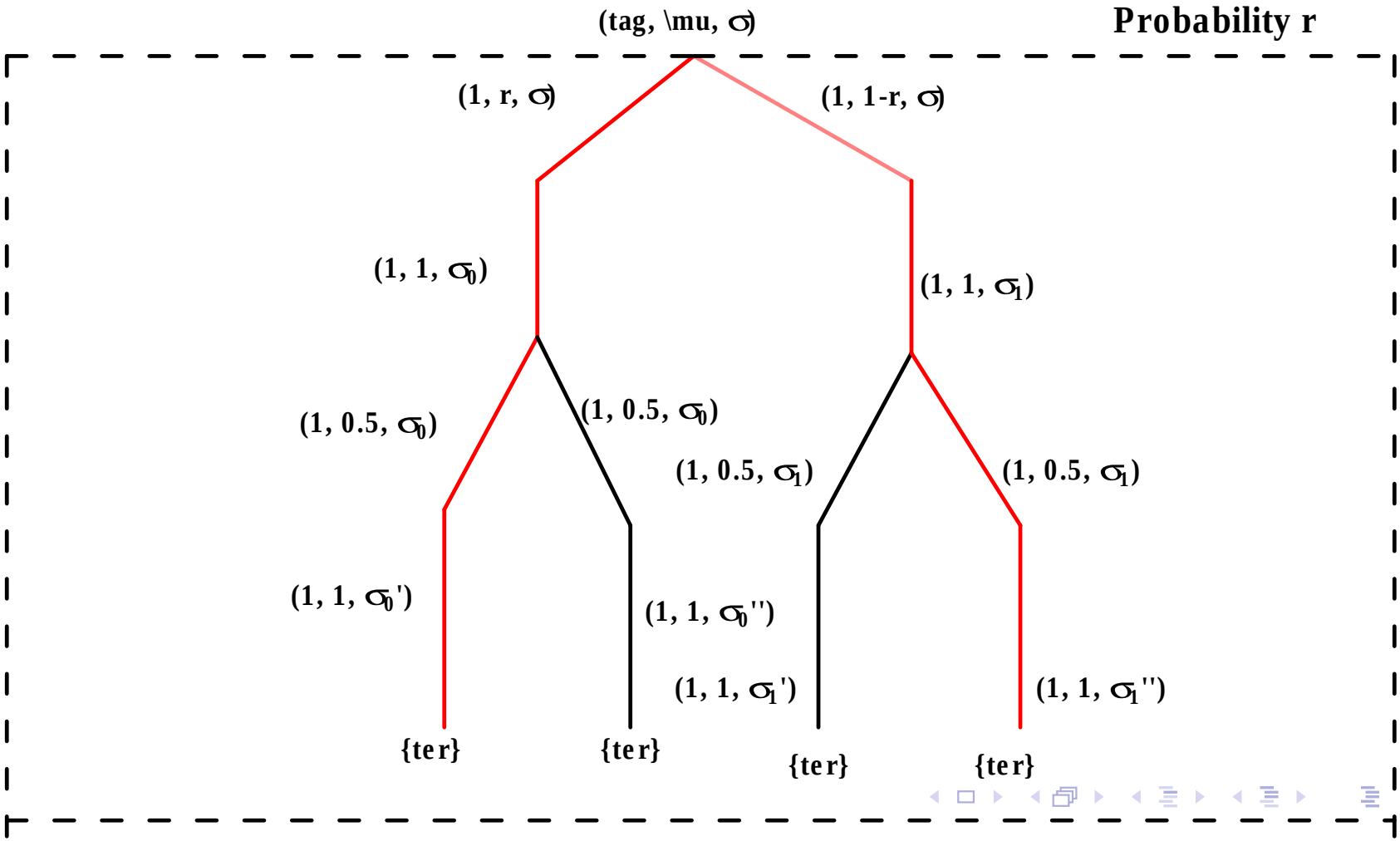
Then, $Prob(P \ ; \ Q, \ x = y)$

$= min\{r \times 1 \times 0.5 \times 1 + (1 - r) \times 1 \times 0.5 \times 1 \mid 0 \le r \le 1\}$

$= min\{0.5 \mid 0 \le r \le 1\} \ = \ 0.5$

Example

Process P;Q



Example

(2) Next we consider the P-tree for program $Q; P$

$$(tag, \mu, \sigma) : \{ \ \{\!| \ (1, 0.5, \sigma) : \{ \ \{\!| \ (1, 1, \sigma_2) : \{ T_{3,r} \mid 0 \leq r \leq 1 \} \ |\!\} \ \}, \\ (1, 0.5, \sigma) : \{ \ \{\!| \ (1, 1, \sigma_3) : \{ T_{4,r} \mid 0 \leq r \leq 1 \} \ |\!\} \ \} \\ |\!\} \ \}$$

where, $T_{3,r} = \{\!| \ (1, r, \sigma_2) : \{\{\!|(1, 1, \sigma_2') : \{ter\}|\!\}\}, \\ (1, 1-r, \sigma_2) : \{\{\!|(1, 1, \sigma_2'') : \{ter\}|\!\}\} \ |\!\}$

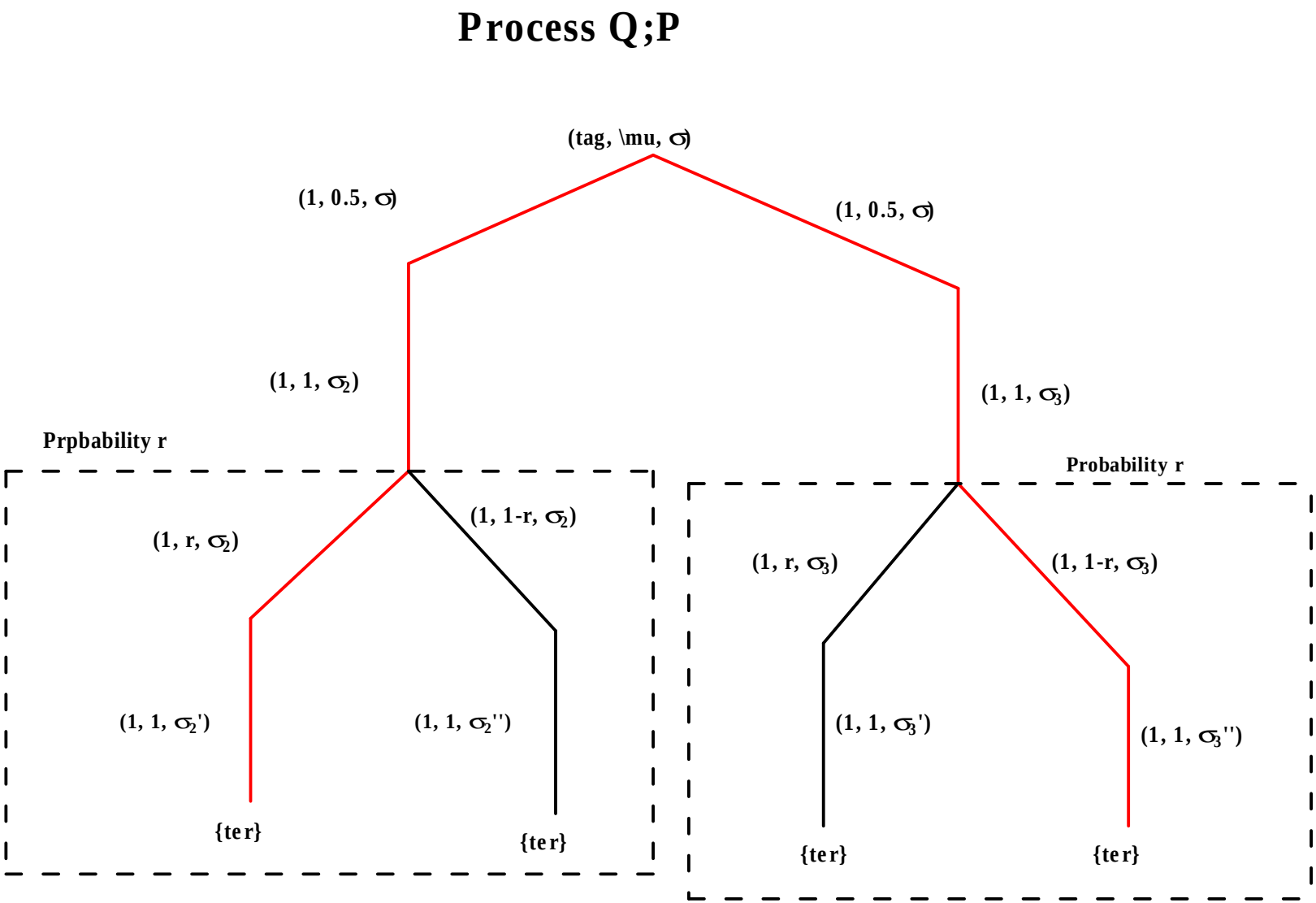
$T_{4,r} = \{\!| \ (1, r, \sigma_3) : \{\{\!|(1, 1, \sigma_3') : \{ter\}|\!\}\}, \\ (1, 1-r, \sigma_3) : \{\{\!|(1, 1, \sigma_3'') : \{ter\}|\!\}\} \ |\!\}$

$\sigma_2 = \{x \mapsto -1, \ y \mapsto 0\}, \quad \sigma_2' = \{x \mapsto 0, \ y \mapsto 0\},$
$\sigma_2'' = \{x \mapsto 1, \ y \mapsto 0\},$
$\sigma_3 = \{x \mapsto -1, \ y \mapsto 1\}, \quad \sigma_3' = \{x \mapsto 0, \ y \mapsto 1\},$
$\sigma_3'' = \{x \mapsto 1, \ y \mapsto 1\},$

Based on the P-tree for program $Q; P$, we can have:

$$Prob(Q \ ; \ P, \ x = y)$$
$$= \ 0.5 \times 1 \times min\{r \times 1 \mid 0 \leq r \leq 1\} \ + \ 0.5 \times 1 \times min\{r \times 1 \mid 0 \leq r \leq 1\}$$
$$= \ 0$$

Example


Process Q;P
(tag, \mu, σ)
(1, 0.5, σ)
(1, 0.5, σ)
(1, 1, σ2)
(1, 1, σ3)
Prpbability r
Probability r
(1, r, σ2)
(1, 1-r, σ2)
(1, r, σ3)
(1, 1-r, σ3)
(1, 1, σ2')
(1, 1, σ2'')
(1, 1, σ3')
(1, 1, σ3'')
{ter}
{ter}
{ter}
{ter}

Outline

Assignment

Assignment. Before the assignment is scheduled, the environment may also have a chance to be scheduled to perform actions.

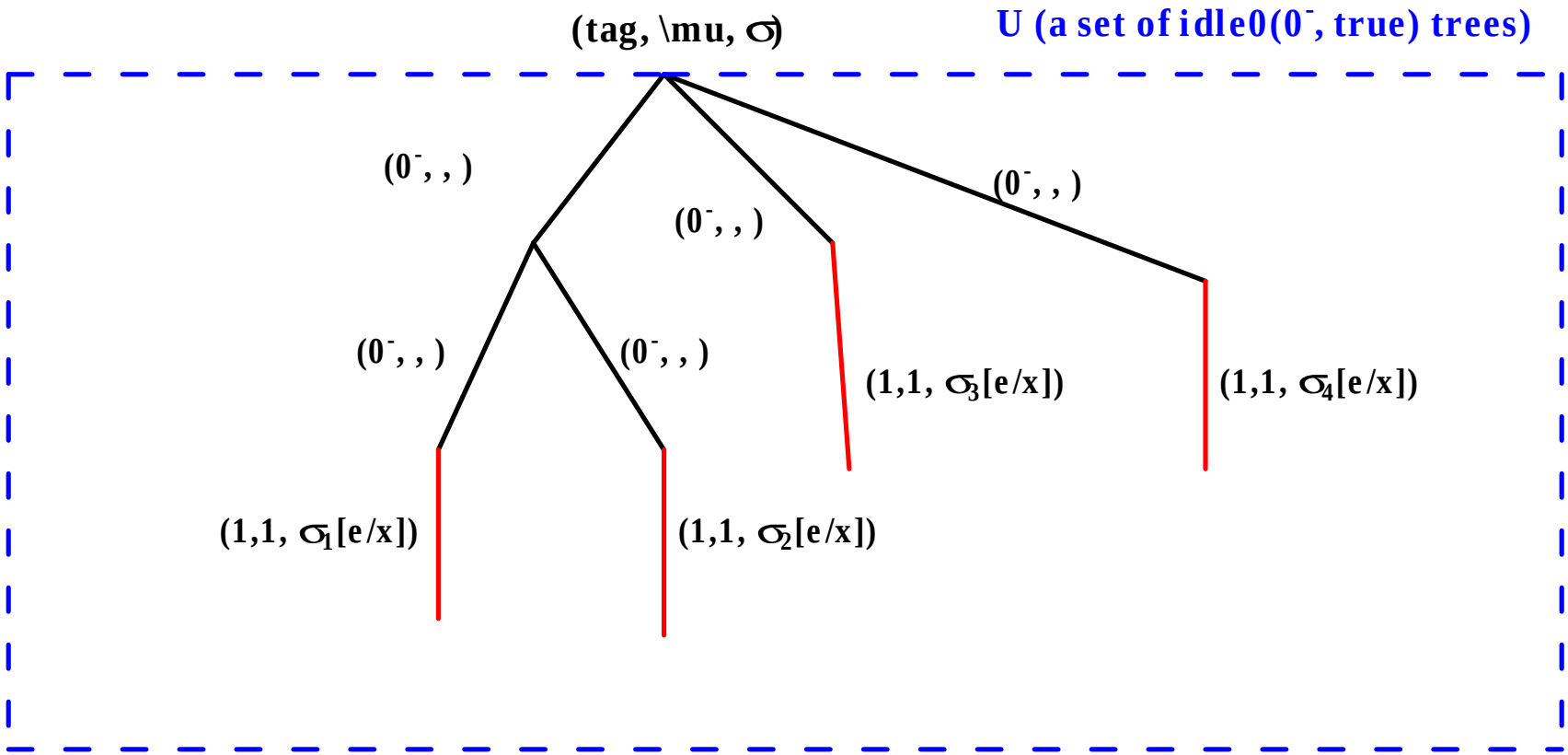
$$
\begin{aligned}
&[\![x := e]\!] \\
=_{df}\ &\{\ \mathit{append}((\mathit{tag}, \mu, \sigma) : U,\ x,\ e) \\
&\qquad\qquad \mid (\mathit{tag}, \mu, \sigma) \in \Sigma \wedge \forall X \in U \bullet X \text{ is } \mathit{idle}0(0^-, \mathit{true})\ \}
\end{aligned}
$$

(1) Formula $\mathit{idle}(0^-, \mathit{true})$ here indicates that, before the assignment is scheduled, the environment can have chances to perform instantaneous actions.

(2) $\mathit{append}((\mathit{tag}, \mu, \sigma) : U,\ x,\ e)$ appends the assignment of variable x with e to the terminating leaf point of each P^--tree in U.

Assignment

The P-tree structure for $x := e$:



Time Delay

The definition for time delay is based on two *tick* and *tick′* functions.

$$tick'((tag, \mu, \sigma) : \{st\})$$
$$=_{df} \begin{cases} (tag, \mu, \sigma) : \{st\} & \text{if } st = wait \text{ or } div \\ (tag, \mu, \sigma) : \{(\surd, \emptyset, \sigma) : \{ter\}\} & \text{if } st = ter \end{cases}$$

and

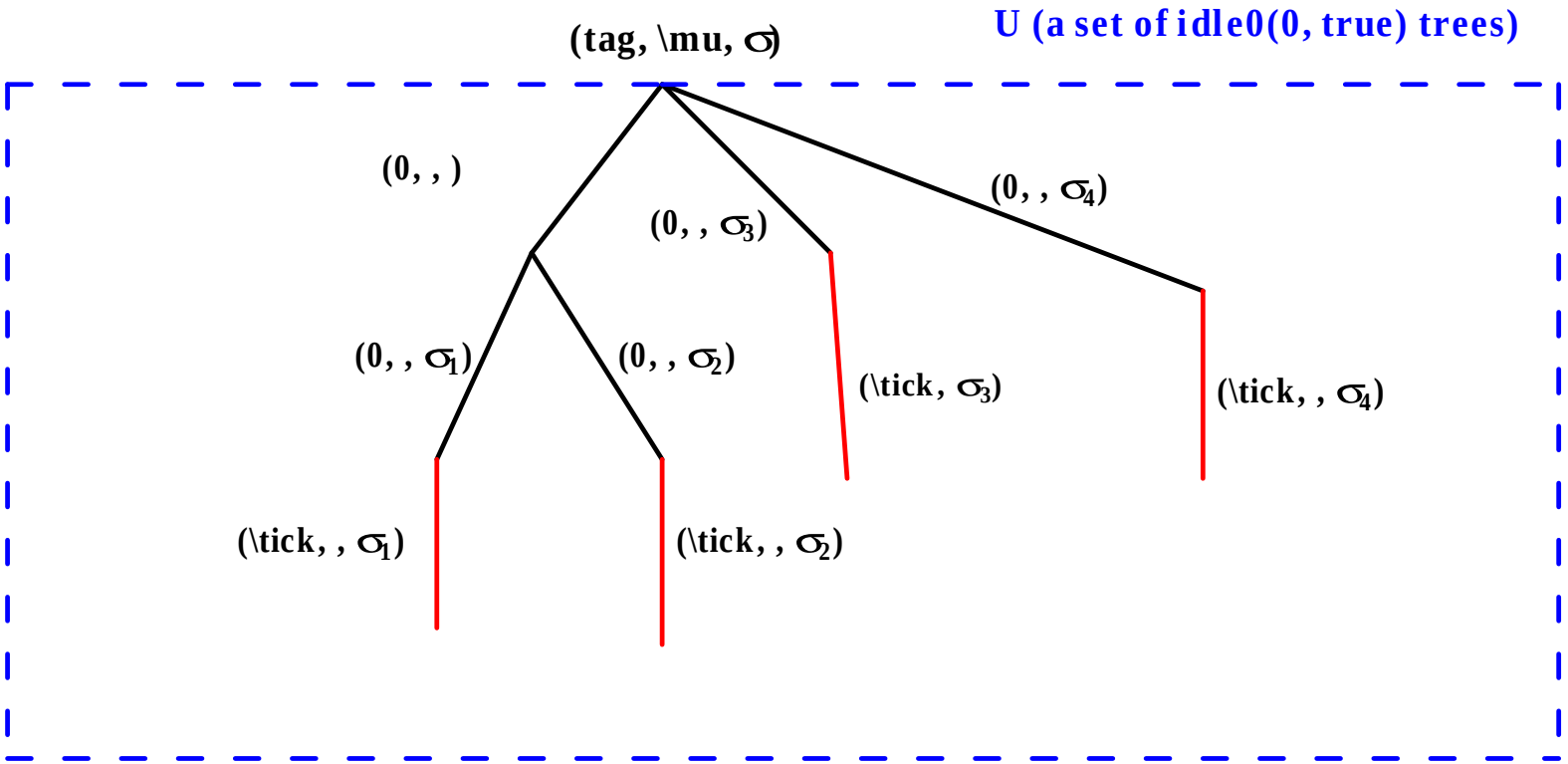
$$tick'((tag, \mu, \sigma) : U) =_{df} (tag, \mu, \sigma) : \{tick(X) \mid X \in U\}$$

Then,

$$[\![\#1]\!] =_{df} \{ (tag, \mu, \sigma) : \{tick(X) \mid X \text{ is } idle0(0, true)\} \mid (tag, \mu, \sigma) \in \Sigma \}$$

$$[\![\#n]\!] =_{df} [\![\#1]\!] ; [\![\#(n-1)]\!]$$

The P-tree structure for #1:



Event Guard

For @b, there are two firing cases:

(1) The first case is that event @b is fired at the initial state, which is denoted as formula $Immefired(b)$.
(2) The second case is that it waits for the environment to fire it. This case can be described using two formulae $Await(b, 1)$ and $Trig(b, 1)$.
(3) Formula $Await(b, 1)$ indicates that all the environment behaviour cannot fire @b. $Trig(b, 1)$ indicates that @b is fired finally.

$$[\![@\, b]\!] =_{df} Immefired(b) \ \cup \ (Await(b, 1) \ ; \ Trig(b, 1))$$

where,

$$\begin{aligned} & Immefired(b) \\ =_{df} & \{ (tag, \mu, \sigma) : \{ter\} \mid (tag, \mu, \sigma) \in \Sigma \wedge b(\sigma) \} \end{aligned}$$

$$\begin{aligned} & Await(b, f) \\ =_{df} & \{ (tag, \mu, \sigma) : U \mid (tag, \mu, \sigma) \in \Sigma \wedge \neg b(\sigma) \wedge \\ & \qquad \forall X \in U \bullet (f = 0 \wedge X \text{ is } idle0(0, \neg b) \ \vee \\ & \qquad\qquad f = 1 \wedge X \text{ is } idle(0, \neg b)) \} \end{aligned}$$

Probabilistic Nondeterminism and Nondeterministic Choice

(1) Probabilistic Nondeterminism $P \sqcap_r Q$

$$[\![P \sqcap_r Q]\!]$$

$$=_{df} \{\, (tag, \mu, \sigma) : U \mid (tag, \mu, \sigma) \in \Sigma \wedge \forall X \in U \bullet X \text{ is } idle0(0^-, true) \,\};$$
$$\{\, (tag, \mu, \sigma) : \{T(r)\} \mid (tag, \mu, \sigma) \in \Sigma \,\}$$

where, $T(r) =_{df} \{\!|\, (1, r, \sigma) : U,\ (1, 1-r, \sigma) : V \,|\!\}$ and

$(tag, \mu, \sigma) : U \in [\![P]\!]$, $(tag, \mu, \sigma) : V \in [\![Q]\!]$

(2) Nondeterministic Choice $P \sqcap Q$

$$[\![P \sqcap Q]\!]$$

$$=_{df} \{\, (tag, \mu, \sigma) : U \mid (tag, \mu, \sigma) \in \Sigma \wedge \forall X \in U \bullet X \text{ is } idle0(0^-, true) \,\};$$
$$\{\, (tag, \mu, \sigma) : \{T(r) \mid 0 \le r \le 1\} \mid (tag, \mu, \sigma) \in \Sigma \,\}$$

Probabilistic Parallel Composition (1)

Probabilistic Merge Operator $\otimes_r$ (based on P^--trees): 16 cases

Case 1: P can perform probabilistic atomic actions initially.

If $P = \{\!|(1, p_i, \sigma_i) : U_i \mid i \in I|\!\}$ and

(1.a) if $Q = \{\!|(1, q_k, \sigma_k) : V_k \mid k \in K|\!\}$, then

$$P \otimes_r Q$$
$$=_{df} \{\!| (1, r \times p_i, \sigma_i) : \{X \otimes_r Q \mid X \in U_i\},$$
$$(1, (1-r) \times p_k, \sigma_k) : \{P \otimes_r Y \mid Y \in V_k\} \mid i \in I \wedge k \in K |\!\}$$

(1.b) if $Q = \{\!|(0^-, q_k, \sigma_k) : V_k \mid k \in K|\!\}$, then $P \otimes_r Q =_{df}$ *undefined*

(1.c) if $Q = \{\!|(0, q_k, \sigma_k) : V_k \mid k \in K|\!\}$, then there exists a permutation j_1, $j_2, \cdots, j_{|I|}$ of K such that $\forall i \in \bullet p_i = q_{j_i}$ and $\sigma_i = \sigma_{j_i}$, and there exists a bijection $f_i : U_i \to V_{j_i}$ such that $\forall X \in U_i \bullet X \otimes_r f_{j_i}(X)$ is well-defined

$$P \otimes_r Q =_{df} \{\!|(1, p_i, \sigma_i) : \{X \otimes_r f_{j_i}(X) \mid X \in U_i\} \mid i \in U|\!\}$$

(1.d) if $Q = \{\!|(\surd, \emptyset, \sigma) : V|\!\}$, then $P \otimes_r Q =_{df}$ *undefined*

Probabilistic Parallel Composition (2)

Based on the definition of $\otimes_r$, now we can give the semantics for probabilistic parallel programs.

$$
\begin{aligned}
& [\![P \parallel_r Q]\!] \\
=_{df} & \{\ (tag, \mu, \sigma) : \{X \otimes_r Y \mid X \in U \ \wedge \ Y \in V \ \wedge \ X \otimes_r Y \text{is well-defined}\} \\
& \qquad \mid (tag, \mu, \sigma) \in \Sigma \ \wedge \ (tag, \mu, \sigma) : U \in [\![P]\!] \ \wedge \\
& \qquad\qquad\qquad\qquad (tag, \mu, \sigma) : V \in [\![Q]\!] \\
& \quad \}
\end{aligned}
$$

General Parallel Composition

The merge operator $\otimes$ (symmetric)

Case 1: P can perform probabilistic atomic actions initially

If $P = \{\!|(1, p_i, \sigma_i) : U_i \mid i \in I|\!\}$ and

(1.a) if $Q = \{\!|(1, q_k, \sigma_k) : V_k\} \mid k \in K|\!\}$, then $P \otimes Q =_{df}$ *undefined*

(1.b) if $Q = \{\!|(tag, q_k, \sigma_k) : V_k\} \mid k \in K|\!\}$ ($tag = 0^-$ or 0), then there exists a permutation $j_1, j_2, \cdots, j_{|I|}$ of K such that $\forall i \in \bullet p_i = q_{j_i}$ and $\sigma_i = \sigma_{j_i}$, and there exists a bijection $f_i : U_i \to V_{j_i}$ such that $\forall X \in U_i \bullet X \otimes f_{j_i}(X)$ is well-defined

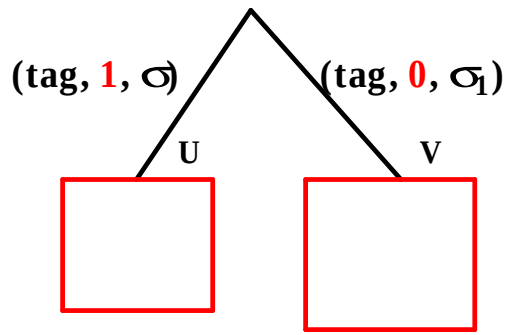
$$P \otimes Q =_{df} \{\!|(1, p_i, \sigma_i) : \{X \otimes f_{j_i}(X) \mid X \in U_i\} \mid i \in I|\!\}$$

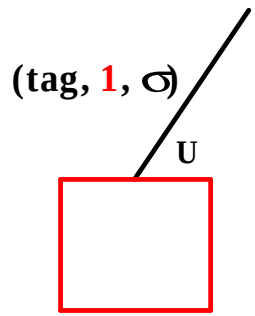
(1.d) if $Q = \{\!|(\surd, \emptyset, \sigma) : V|\!\}$, then $P \otimes Q =_{df}$ *undefined*

Program Equivalence $\approx$

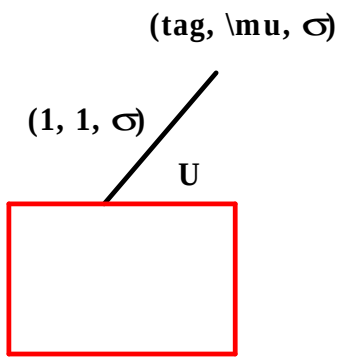
In order to consider program equivalence, we need to study the equivalence between P-trees (or P^{-}-trees).

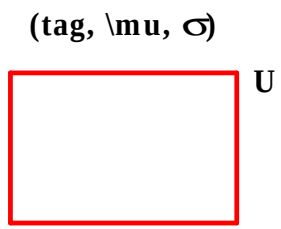
Case 1: $\{\!|(tag, 1, \sigma) : U,\ (tag, 0, \sigma) : V\}\!\}$ and $\{\!|(tag, 1, \sigma) : U|\!\}$





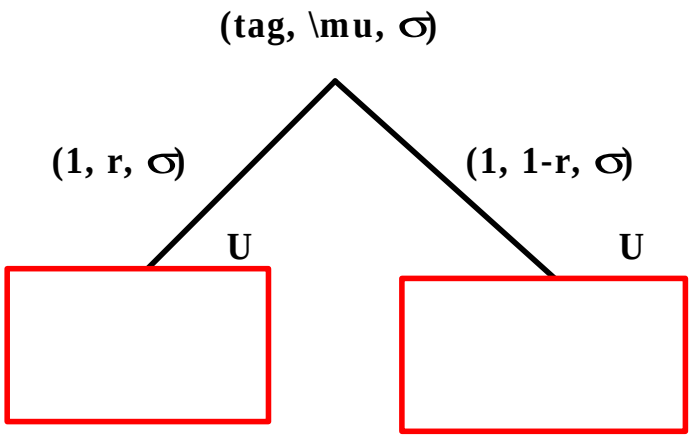
Case 2: $(tag, \mu, \sigma) : \{\{\!|(1, 1, \sigma) : U|\!\}\}$ and $(tag, \mu, \sigma) : U$

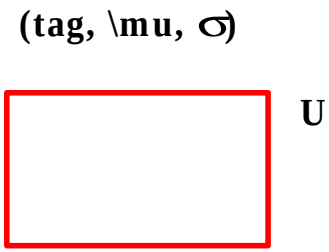




Program Equivalence $\approx$

Case 3: $(tag, \mu, \sigma) : \{\!|(1, r, \sigma) : U,\ (1, 1-r, \sigma) : U|\!\}$ and $(tag, \mu, \sigma) : U$





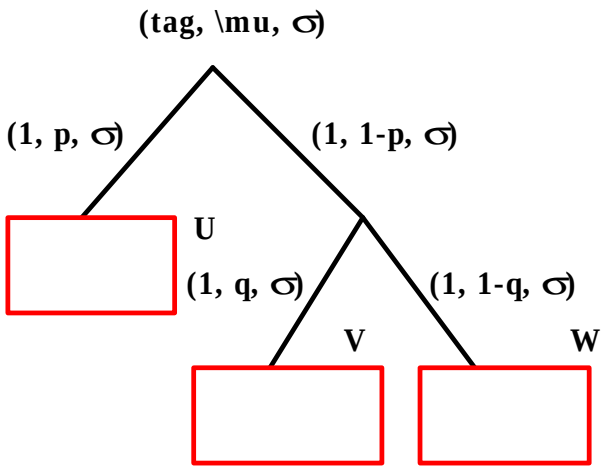
Case 4:

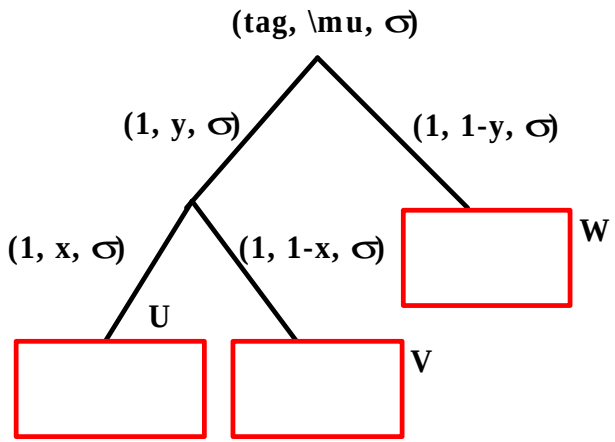
$$(tag, \mu, \sigma) : \{\{\!|(1, p, \sigma) : U,$$
$$(1, 1-p, \sigma) : \{\{\!|(1, q, \sigma) : V, (1, 1-q, \sigma) : W|\!\}\}|\!\}\}$$

and

$$(tag, \mu, \sigma) : \{\{\!|(1, y, \sigma) : \{\{\!|(1, x, \sigma) : U, (1, 1-x, \sigma) : V\}|\!\}\},$$
$$(1, 1-y, \sigma) : W\}|\!\}\}$$

where, $x = p/(p + q - p \times q)$, $y = p + q - p \times q$





Outline

Algebraic Laws (1)

(delay-1) $\#n; \#m \approx \#(n+m)$

(prob-1) $P \sqcap_p P \approx P$

(prob-2) $P \sqcap_{p_1} Q \approx Q \sqcap_{1-p_1} P$

(prob-3) $P \sqcap_p (Q \sqcap_q R) \approx (P \sqcap_x Q) \sqcap_y R$

where $x = p/(p+q-p\times q)$ and

$y = p+q-p\times q$

Algebraic Laws (2)

(par-3-1) Let

$$P = \ [\!]_{i \in I}\{[p_i]\ choice_{j \in J_i}(b_{ij} \& (x_{ij} := e_{ij})\, P_{ij})\} \quad \text{and}$$

$$Q = \ [\!]_{k \in K}\{[q_k]\ choice_{l \in L_k}(b_{kl} \& (x_{kl} := e_{kl})\, P_{kl})\}$$

Then

$$P \parallel_r Q$$

$$\approx \ [\!]_{i \in I}\{[r \times p_i]\ choice_{j \in J_i}(b_{ij} \& (x_{ij} := e_{ij})\, P_{ij} \parallel_r Q\}$$

$$[\!][\!]_{k \in K}\{[(1-r) \times q_k]\ choice_{l \in L_k}(b_{kl} \& (x_{kl} := e_{kl})\, P \parallel_r Q_{kl}\}$$

and

$$P \parallel Q$$

$$\approx \quad [\!]_{i \in I}\{[p_i]\ choice_{j \in J_i}(b_{ij} \& (x_{ij} := e_{ij})\, P_{ij} \parallel Q\}$$

$$\textbf{or}\ [\!]_{k \in K}\{[q_k]\ choice_{l \in L_k}(b_{kl} \& (x_{kl} := e_{kl})\, P \parallel Q_{kl}\}$$

Algebraic Laws (3)

(par-3-6) Let $P = \|_{i \in I}\{@b_i\ P_i\}$ and $Q = \|_{j \in J}\{@c_j\ Q_j\}$

Then

$$
\begin{aligned}
& P \odot Q \\
\approx\ & \|_{i \in I}\{@(b_i \wedge \neg c)\ P_i \odot Q\} \| \|_{j \in J}\{@(c_j \wedge \neg b)\ P \odot Q_j\} \\
& \| \|_{i \in I \wedge j \in J}\{@(b_i \wedge c_j)\ P_i \odot Q_j\}
\end{aligned}
$$

where, $\odot \in \{\|_r,\ \|\}$,

$$b = \vee_{i \in I}\, b_i \quad \text{and} \quad c = \vee_{j \in J}\, c_j$$

Outline

Conclusion and Future Work

(1) **Conclusion**

(a) We studied the denotational semantics for *PTSC*.

(b) We introduced the concept of P-tree (and P^{-}-tree).

(c) In order to deal with program equivalence based on the achieved denotational semantics, we defined a set of flattening relations.

(d) We have explored a set of algebraic laws for *PTSC*.

(2) **Future Work**

(a) We would like to link the denotational semantics with operational semantics and algebraic semantics respectively.

(b) The deduction approach for *PTSC* is also challenging to work on.

Thank you very much!