

Preserving (Security) Properties under Action Refinement

Annalisa Bossi, Carla Piazza, and Sabina Rossi

Dipartimento di Informatica
Università Ca' Foscari di Venezia

{bossi,piazza,srossi}@dsi.unive.it

CILC 2004 - Parma, June 2004

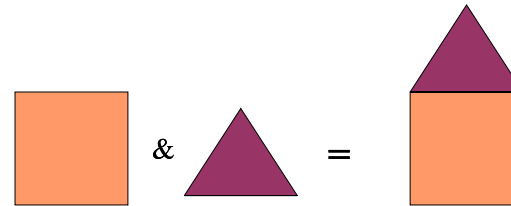
Protect Confidential Data in a Multilevel System

- ▷ **Information Flow Security** aims at guaranteeing that no **high** level (**confidential**) information is revealed to users at **low** level, even in the presence of any possible **malicious process**
- ▷ **Non-Interference** [Goguen-Meseguer'82]: **information does not flow** from high to low if the **high behavior** has **no effect** on what **low** level can **observe**
- ▷ **Development of Complex Systems**: it is important to **preserve the security** properties of interest during the **development steps**

Development of Complex Systems

The systematic development of complex systems usually relies on

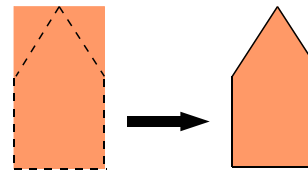
- ▷ **Composition:** building blocks are put together (e.g., parallel composition)



The composition of secure parts has to be secure as a whole

Compositional Non-Interference properties have been studied

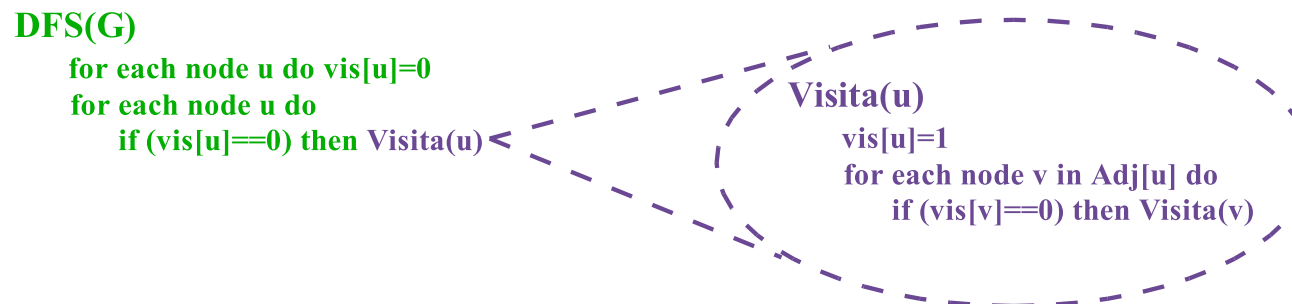
- ▷ **Refinement:** abstract specifications are refined into more concrete ones



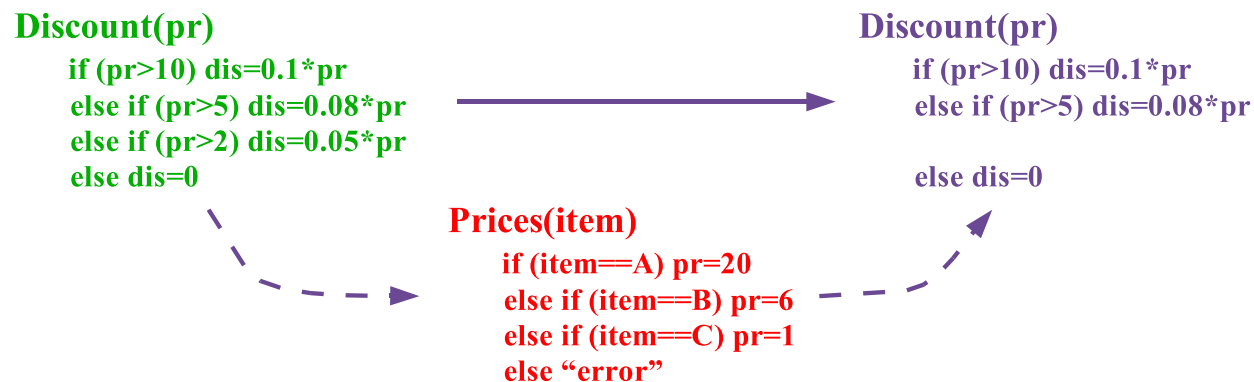
Non-Interference properties based on sets of execution sequences are hard to preserve under refinement

Vertical and Horizontal Refinement

- ▷ **Vertical Refinement (Action Refinement)**: abstract-level primitives (actions) are expanded into concrete-level implementations (macros)



- ▷ **Horizontal Refinement**: abstract specifications are refined into more concrete ones by removing undesired behaviors



Refinement and Security

In our work we consider **Action Refinement** and we

- ▷ introduce a syntactic notion of **refinement** based on **context composition** instead of **sequential operator** “ ; ”
- ▷ analyze under which conditions our notion of refinement **preserves information flow security** properties. In particular, we consider properties defined through **unwinding conditions**

Plan of the Talk

- ▷ The SPA language: syntax and semantics
- ▷ The notion of Refinement and compositionality
- ▷ Security as Unwinding conditions
- ▷ Refinements preserving Security
- ▷ Comparisons
- ▷ Conclusions

The SPA syntax

E	$::=$	$\mathbf{0}$	<i>empty process</i>
		Z	<i>variable</i>
		$a.E$	<i>prefix</i>
		$E + E$	<i>non-det. choice</i>
		$E \mid E$	<i>parallel composition</i>
		$E \setminus v$	<i>restriction</i>
		$E[f]$	<i>relabelling</i>
		$recZ.T$	<i>recursion</i>

- ▷ processes have **not free variables**
- contexts can have **free variables**
- ▷ **H high** actions and **L low** actions

The SPA semantics – I

Semantics given through transition relations

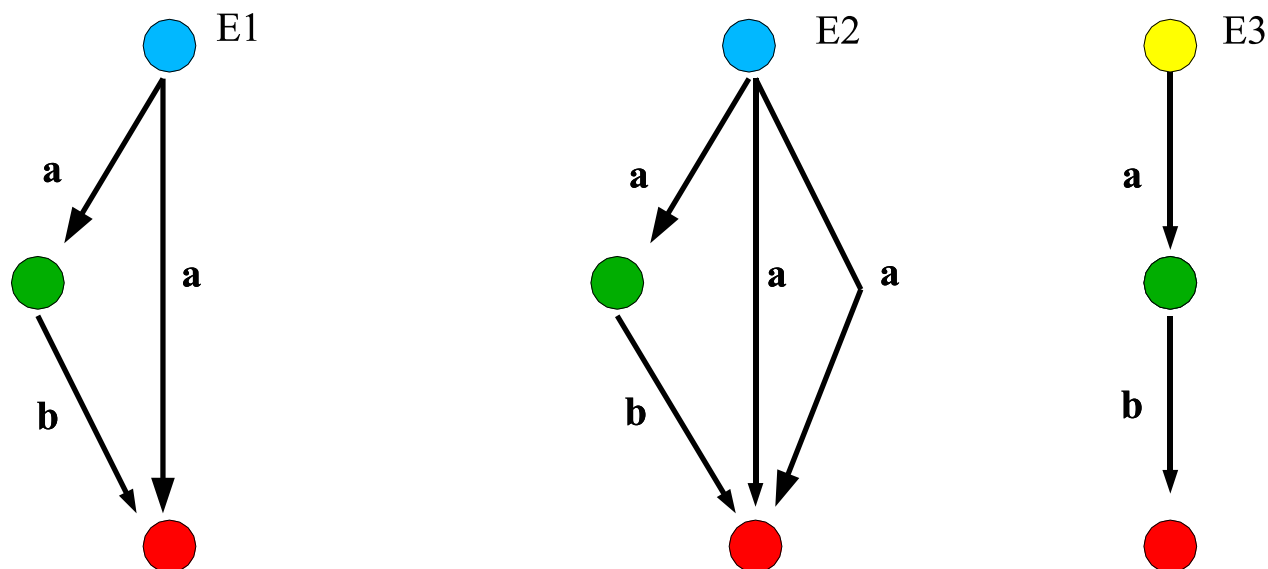
Input		Output	
	$a.E \xrightarrow{a} E$		$a.E \xrightarrow{\bar{a}} E$
Parallel	$E_1 \xrightarrow{a} E'_1$		$E_1 \xrightarrow{a} E'_1 \quad E_2 \xrightarrow{\bar{a}} E'_2$
	$E_1 E_2 \xrightarrow{a} E'_1 E_2$		$E_1 E_2 \xrightarrow{\tau} E'_1 E'_2$

Behavioral equivalences establish equalities among processes

The SPA semantics – II

▷ Labelled Transition Systems (LTS) represent processes

▷ $E1 = a.b.0 + a.0$ $E2 = a.b.0 + a.0 + a.0$ $E3 = a.b.0$

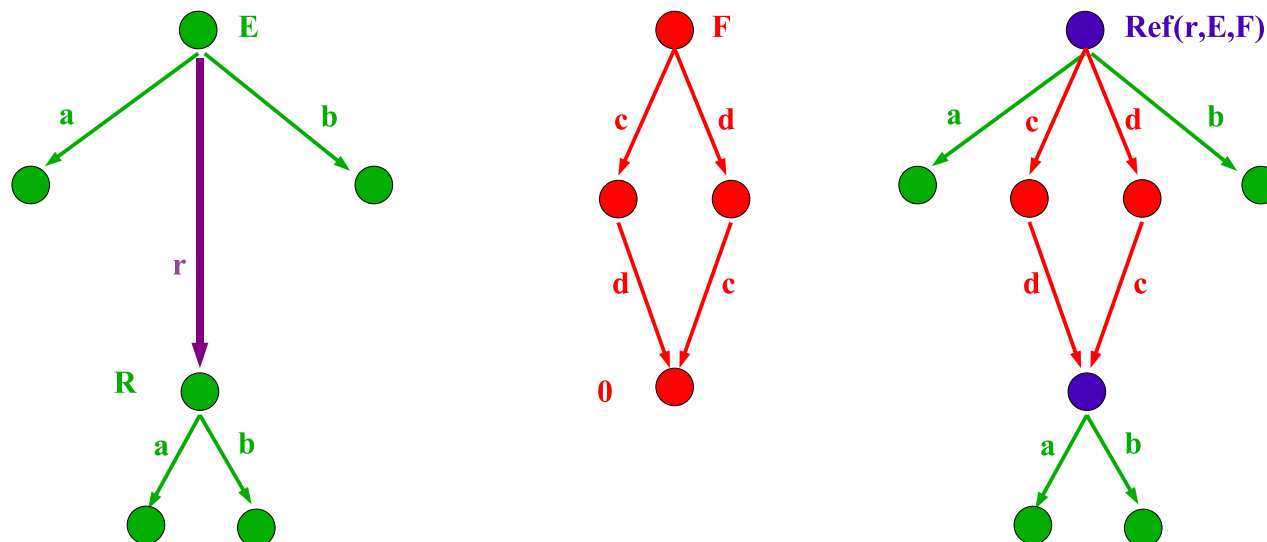


▷ $E1$ and $E2$ are “equivalent”.

▷ $E3$ cannot “simulate” the rightmost a of $E1$!

Refinement – Intuition

We intend to replace an action r occurring in E with a process F



- ▷ find the subprocess(es) of E which follows $r \Rightarrow R$
- ▷ create holes in F in correspondence of $0 \Rightarrow F^0[Y]$
- ▷ replace $r.R$ in E with $F^0[R]$

Refinement – Formalization

The refinement of r in E with F is the process $Ref(r, E, F)$:

$$Ref(r, r.E_1, F) \equiv F^0[Ref(r, E_1, F)]$$

$$Ref(r, \mathbf{0}, F) \equiv \mathbf{0}$$

$$Ref(r, Z, F) \equiv Z$$

$$Ref(r, a.E_1, F) \equiv a.Ref(r, E_1, F), \text{ if } a \neq r$$

$$Ref(r, E_1 \setminus v, F) \equiv Ref(r, E_1, F) \setminus v$$

$$Ref(r, E_1 + E_2, F) \equiv Ref(r, E_1, F) + Ref(r, E_2, F)$$

$$Ref(r, recZ.E_1, F) \equiv recZ.Ref(r, E_1, F)$$

...

Refinability Conditions

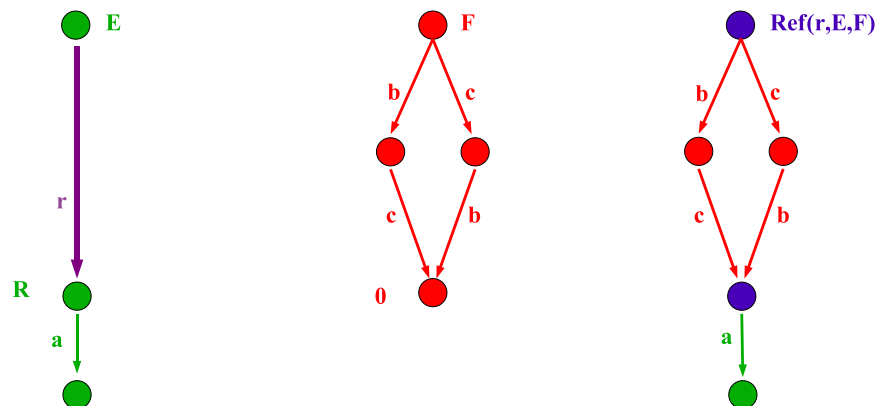
The syntactic definition corresponds to the intuitive one under some conditions on **relabellings**, **restrictions**, and **parallel composition**

Let $E \equiv r.a.0$ and $F \equiv b.0|c.0$

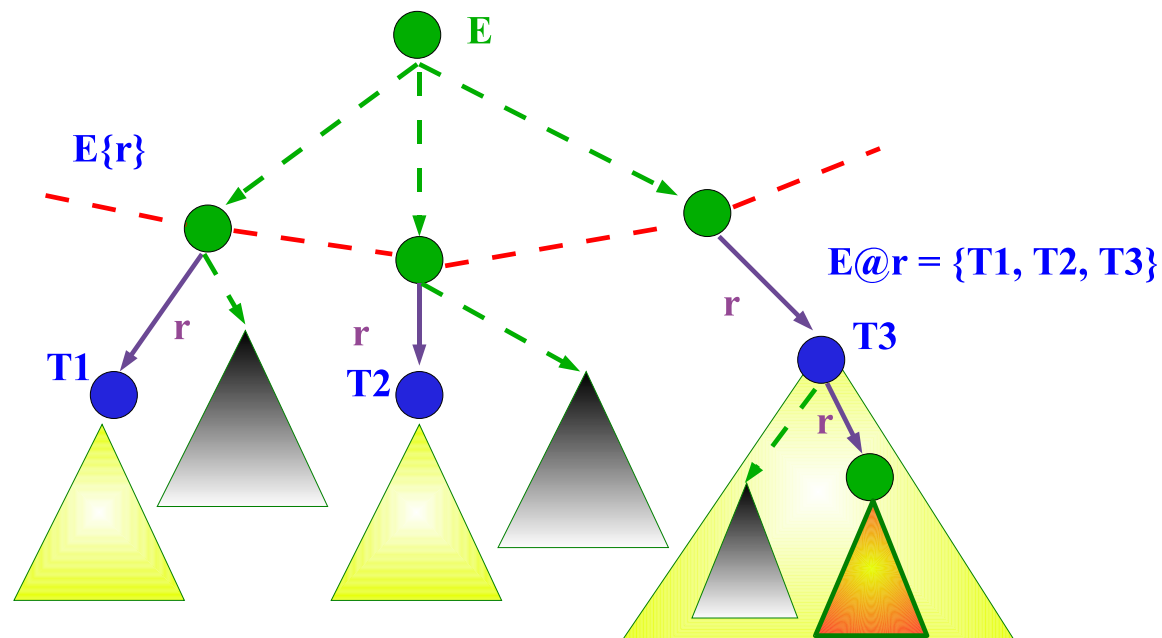
Applying the syntactic definition we get

$$Ref(r, E, F) \equiv b.a.0|c.a.0 \equiv b.a.c.a.0 + \dots + c.a.b.a.0$$

while we would expect $(b.0|c.0); a.0 \equiv b.c.a.0 + c.b.a.0$, i.e.,



A Characterization of Ref : $E@r$ and $E\{r\}$



$$ParRef(r, E, F) = E\{r\}[F^0[T_1], F^0[T_2], F^0[T_3]]$$

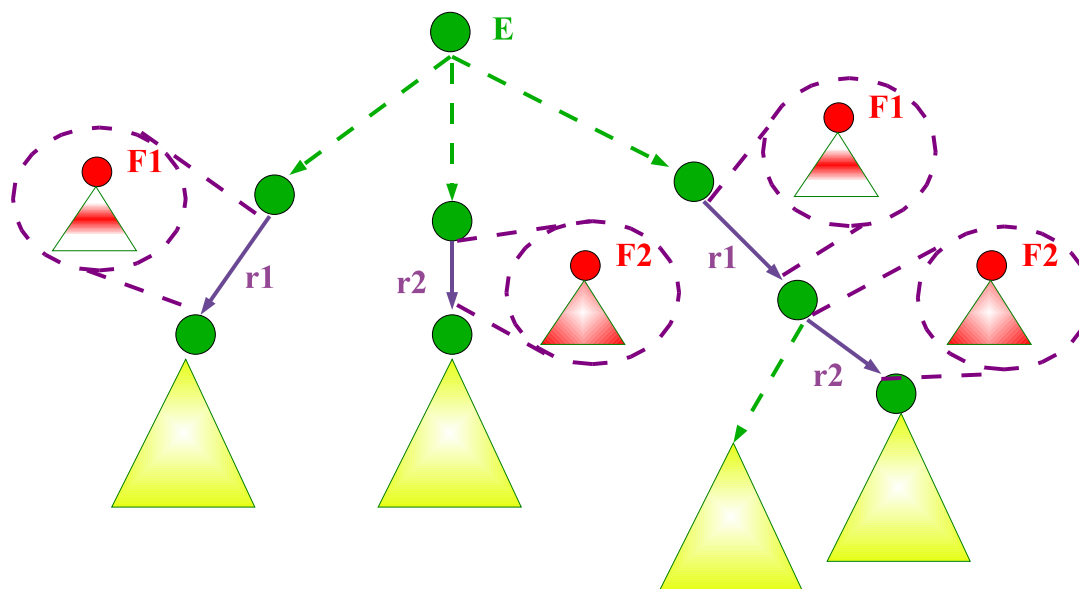
$$Ref(r, E, F) = ParRef^n(r, E, F)$$

Refinement and Compositionality

If F_1 and F_2 have no occurrences of $r_1, r_2, \overline{r_1}, \overline{r_2}$ then

$$Ref(r_2, Ref(r_1, E, F_1), F_2) \equiv Ref(r_1, Ref(r_2, E, F_2), F_1)$$

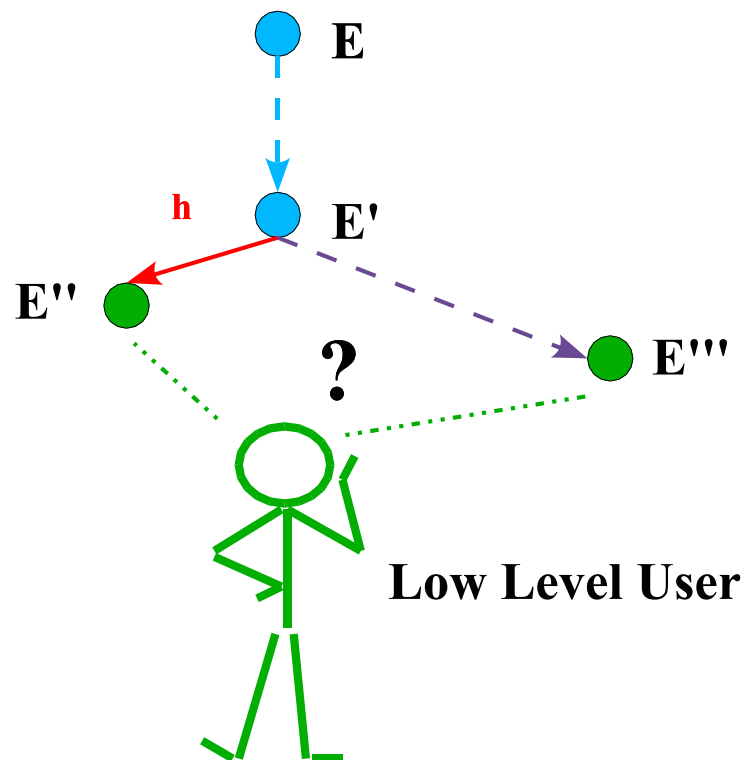
i.e., the order does not matter



If F_2 has no occurrences of $r_1, \overline{r_1}$ then $Ref(r_2, Ref(r_1, E, F_1), F_2) \equiv Ref(r_1, Ref(r_2, E, F_2), Ref(r_2, F_1, F_2))$

Security as Unwinding - Intuition

If the **high** level user can perform h reaching E'' from E' , then also E'' is **reachable** from E' and E'' and E'' are undistinguishable for the **low** level user



Many security properties are instances of this scheme: **P_BNDC**, **SBNDC**, **CP_BNDC**, **SNDC**

Security as Unwinding - Formalization

Let \sim^l be a low level observational equivalence

Let $---\rightarrow$ be a reachability relation

Generalized Unwinding

$$\mathcal{W}(\sim^l, ---\rightarrow) = \{E \in \mathcal{E} \mid \forall F, G \in Reach(E), \text{ if } F \xrightarrow{h} G \text{ then} \\ \exists G' \text{ such that } F ---\rightarrow G' \text{ and } G \sim^l G'\}$$

Refinement and Security

Definition

Let $s \in (L \cup \{\tau\})^*$ be such that $E \xrightarrow{s} E'$ entails $E \dashrightarrow E'$

The class $\mathcal{C}(\dashrightarrow)$ is the minimum class closed w.r.t.:

0 Z $l.C_1$ $h.C_1 + s.C_1$ $C_1 + C_2$ $C_1 \setminus v$ $C_1[f]$ $C_1|C_2$ $recZ.C_1$

with $s \cap v = \emptyset$ and $f[s] = s$

Theorem

Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding condition.

If E and F^0 are in $\mathcal{C}(\dashrightarrow)$, then

$$Ref(r, E, F) \in \mathcal{W}(\sim^l, \dashrightarrow) \cap \mathcal{C}(\dashrightarrow)$$

Comparisons

Our refinement is strongly related to the refinement introduced by Aceto and Hennessy in Information and Computation 115(2):

- ▷ they use the sequential operator ; and add a notion of termination ✓ to the semantics
- ▷ in their refinement actions are syntactically replaced by refining processes

Our refinement is equivalent to their refinement, without using ; and ✓

Since we do not change the language we do not have to modify the unwinding security framework

Conclusions

- ▷ We presented a notion of **Action Refinement**
- ▷ We showed that it satisfies nice **compositionality properties**
- ▷ We analyzed how to preserve **unwinding based security properties** under **action refinement**

ESTENDERE