

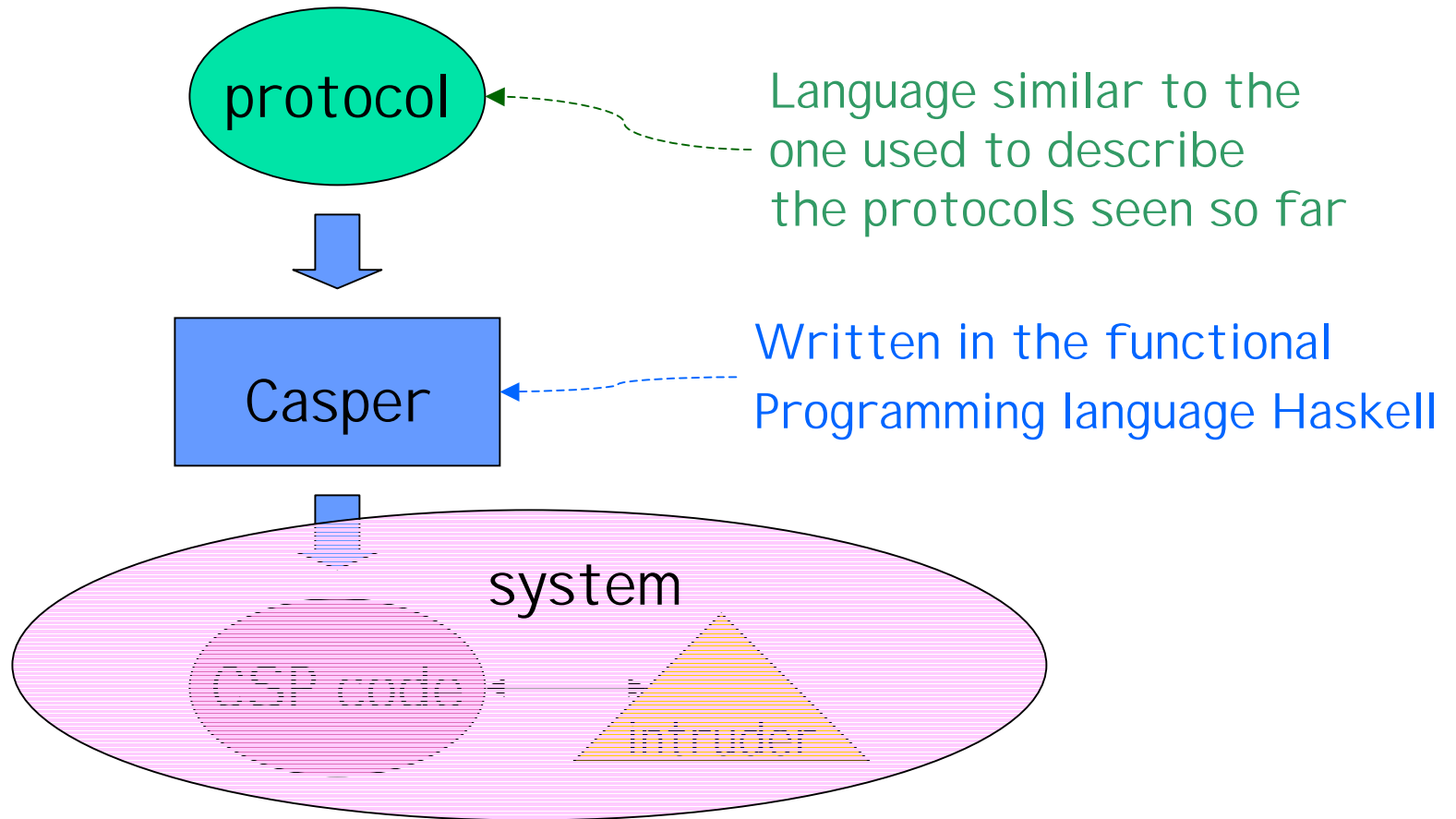


CSP tools for verification of Sec Prot

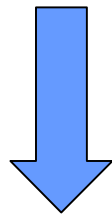
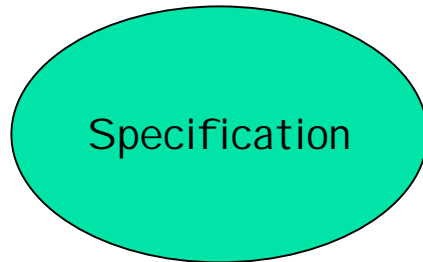
- Overview of the lecture
 - The Casper interface
 - Refinement checking and FDR
 - Model checking
 - Theorem proving
 - Safe simplifying transformations

The Casper interface

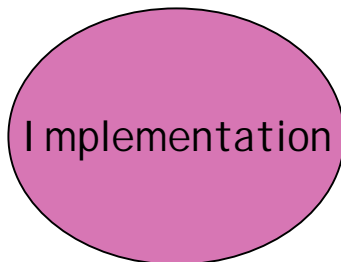
- Casper is essentially a compiler



Refinement checking and FDR



Refinement



Specification

An abstract description of the protocol,
where properties are easy to check

Refinement

A transformation preserving the properties
Usually this means that the implementation
must be **less nondeterministic** than the
specification.

Question: *why?*

Answer: *the properties usually are universal:
they must be valid in all runs*

Implementation

A formal description of the real system and
its components



Refinement checking and FDR

- The notion of refinement
 - The implementation should be a **refinement** of the specification, in the sense of preserving its properties. Hence, less nondeterministic
 - **Refinement checking**: checking that the implementation is indeed a refinement of the specification
 - Obviously, the notion of refinement depends on the **intended semantic**. In CSP traditionally we consider 3 kinds of semantics:
 - Traces
 - Failures
 - Failures and Divergences



Refinement checking and FDR

■ Traces

- The set of the sequences of visible actions in all possible runs

- Example:

$A = a \rightarrow b \rightarrow c \rightarrow \text{Stop} [] b \rightarrow c \rightarrow a \rightarrow \text{Stop}$

$\text{Tr}(A) = \{ a.b.c, b.c.a \}$

$B = a \rightarrow \text{Stop} ||| b \rightarrow c \rightarrow \text{Stop}$

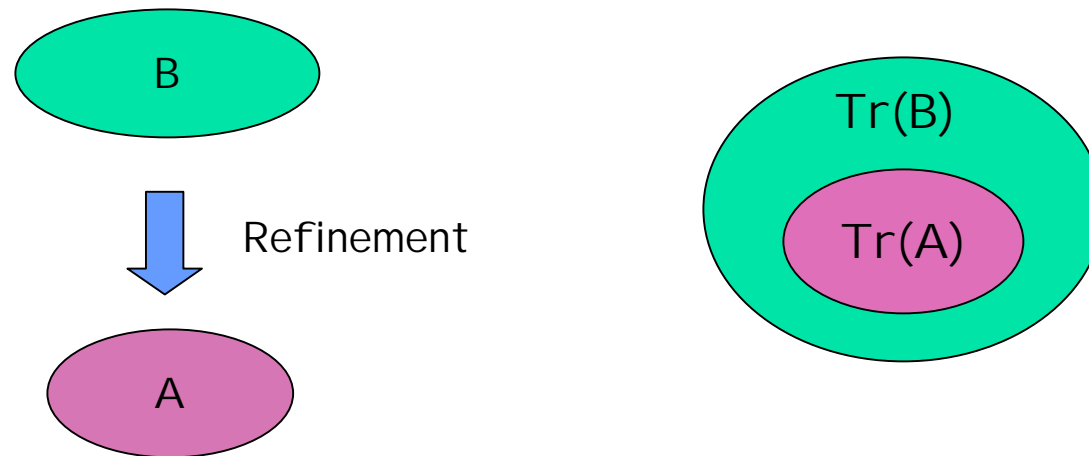
$\text{Tr}(A) = \{ a.b.c, b.a.c, b.c.a \}$

Note that A is less nondeterministic than B :

A represents only a subset of the possible runs of B

Refinement checking and FDR

- We want to define formally the notion of **refinement** wrt trace semantics, in such a way that it captures the concept of “**less nondeterministic**”.
- **Question:** what should be the formal definition of “**A** is a refinement of **B**” in terms of their traces?
- **Answer:** $\text{Tr}(A)$ is a subset of $\text{Tr}(B)$



Refinement checking and FDR

■ Limits of trace semantics:

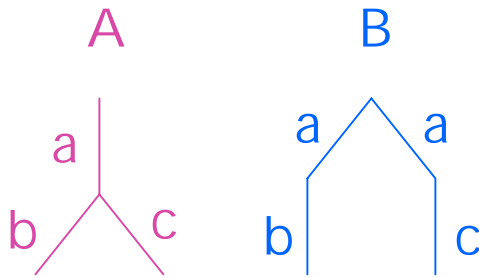
It is **not compositional**. Namely, trace refinement is not preserved under contexts.

This means that the analysis based on traces cannot be performed modularly: it must be performed on the whole system at once.

■ Example

■ $A = a \rightarrow (b \rightarrow \text{Stop} [] c \rightarrow \text{Stop})$

■ $B = a \rightarrow \text{Stop} [] a \rightarrow c \rightarrow \text{Stop}$



$\text{Tr}(A) =$
 $\text{Tr}(B) =$
 $\{ a.b, a.c \}$

Context

$C = a \rightarrow b \rightarrow \text{Stop}$

$\text{Tr}(C || A) = \{ a.b \}$

\neq

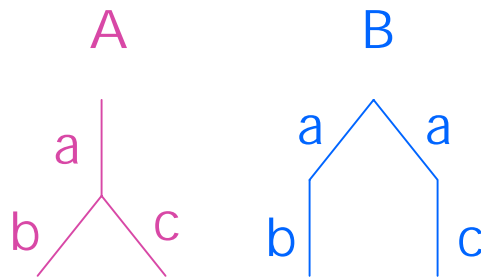
$\text{Tr}(C || B) = \{ a.b, a \}$

Refinement checking and FDR

- **Stronger, compositional semantics**

- **Failures:** this semantics encodes the branching structure of a process by representing, after each partial trace, the set of actions that are not allowed

- Example:



$$F(A) = \{ \langle a, \{a\} \rangle, \dots \}$$

\neq

$$F(B) = \{ \langle a, \{a,b\} \rangle, \langle a, \{a,c\} \rangle, \dots \}$$

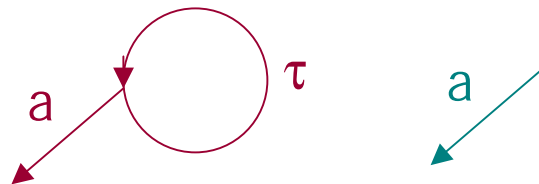
- The notion of refinement is defined in such a way that **A** is a refinement of **B**, but not vice versa
- This notion of refinement is preserved under closure on possible contexts

Refinement checking and FDR

- Stronger, compositional semantics

- **Failures and Divergences:** This semantics encodes all the information of Failure semantics, plus additional information about divergences (**livelocks**). As such, it is suitable also for the analysis of **liveness** properties (not only **safety**)

- Example:



These two processes
Have the same failures
but different divergences

- It is considered the standard model of CSP
- **FDR** means: Failure and Divergences Refinement
 - Automatic refinement check for finite state systems (they can have infinite computations, but the number of states is finite)



Model checking

- **Model checking basics**
- The security property we want to verify is expressed as a temporal logic formula F
- The protocol is expressed as a system S of processes (the processes involved with the protocol and, possibly, the intruder)
- S is represented as a labeled transition system T , which is seen as an interpretation of F
- Model checking consists in showing that T is a model of F , namely that T validates F
 - Always possible with Finite State systems
 - Also possible for some restricted cases of Infinite State systems

Model checking

Example

$A = a \rightarrow b \rightarrow A$

$B = c \rightarrow d \rightarrow B$

$S = A ||| B$

$A = a \rightarrow A1 \quad A1 = b \rightarrow A$

$B = c \rightarrow B1 \quad B1 = d \rightarrow B$

$S = A ||| B \quad S1 = A ||| B1$

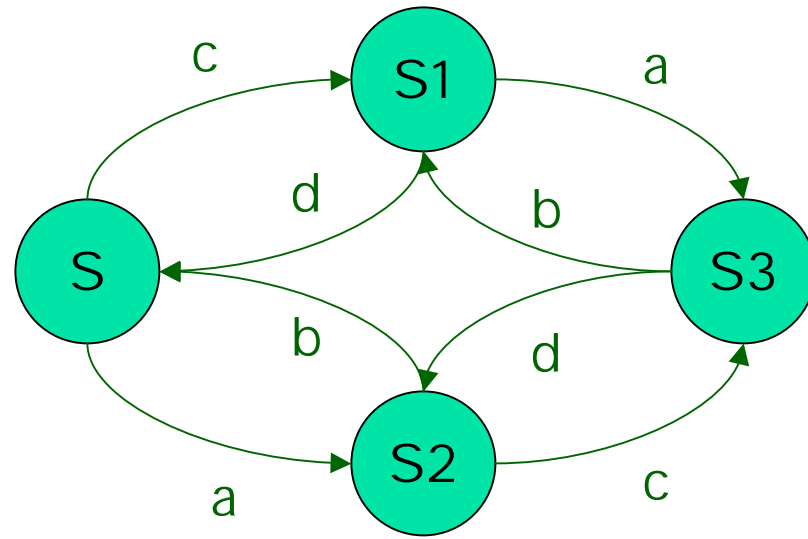
$S2 = A1 ||| B \quad S3 = A1 ||| B1$

- Temporal operators:

- F : forever
- E : eventually

- Some formulae

- $F E (a \text{ or } c)$ True
- $F E a$ False
- $E F (a \text{ or } c)$ False





Theorem proving

- Based on a set of inference rules that model a satisfiability relation $S \text{ sat } P$

- Example: parallel rule

$$\frac{\text{Forall } i. (S_i \text{ sat } (R \text{ precedes } T))}{(|||_i S_i) \text{ sat } (R \text{ precedes } T)}$$

- Properties of the inference system:
 - Semiautomatic (invariant needed for recursive definitions)
 - Sound and relatively complete
 - All the properties of the Yahalom protocol seen in previous lecture can be easily verified using this system (see proofs in the book of Ryan and Schneider)