Formal Security Analysis with Interacting State Machines

David von Oheimb and Volkmar Lotz
Siemens AG, Corporate Technology, D-81730 Munich
{David.von.Oheimb,Volkmar.Lotz}@siemens.com

Overview

- Introduction
- ISM framework
- Case Studies
  - Infineon SLE 66
  - Needham-Schroeder
- Conclusion

Motivation

Our customers: IT developers with security concerns

- Requirements analysis for security, e.g. Siemens Med: syngo
- Evaluation according to ITSEC and CC, e.g. Infineon SLE 66/88

Our mission: Rigorous security analysis

- security modeling and verification using formal methods
- checks and presentation with machine assistance

The question: Which framework shall we employ?

Requirements

Expressiveness: state transformation, concurrency, message passing

Flexibility: adaptation and extension

Simplicity: minimal expertise and time

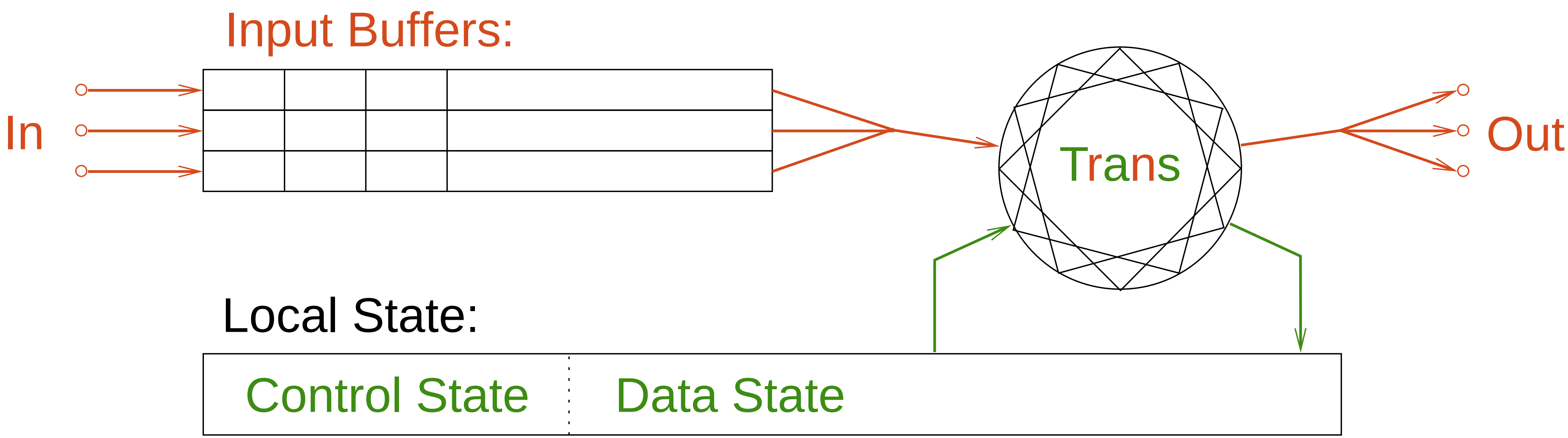
Maturity of the semantics $\leadsto$ refinement, compositionality

Graphical capabilities: overview and intuition

Tool support: mature and freely available (including sources)

Interacting State Machines (ISMs)

Automaton with (nondeterministic) state transitions +
Buffered i/o simultaneously on multiple connections



Transitions in executable and/or property-oriented style
Finite executions only ($\leadsto$ no liveness properties)

ISM Framework

AutoFocus: graphical specification (and simulation) tool

Syntactic perspective

Graphical documentation

Type and consistency checks

Isabelle/HOL: powerful interactive theorem prover

Semantic perspective

Textual documentation

Validation and correctness proofs

AutoFocus drawing $\longrightarrow$ Quest file $\xrightarrow{\mathrm{Conv}_1}$ Isabelle theory file

Within Isabelle: **ism** sections $\xrightarrow{\mathrm{Conv}_2}$ Standard HOL definitions

LKW Model for the Infineon SLE66

One of first formal models for security properties of hardware

Extrinsic value: Security certification on level ITSEC E4 / CC EAL5

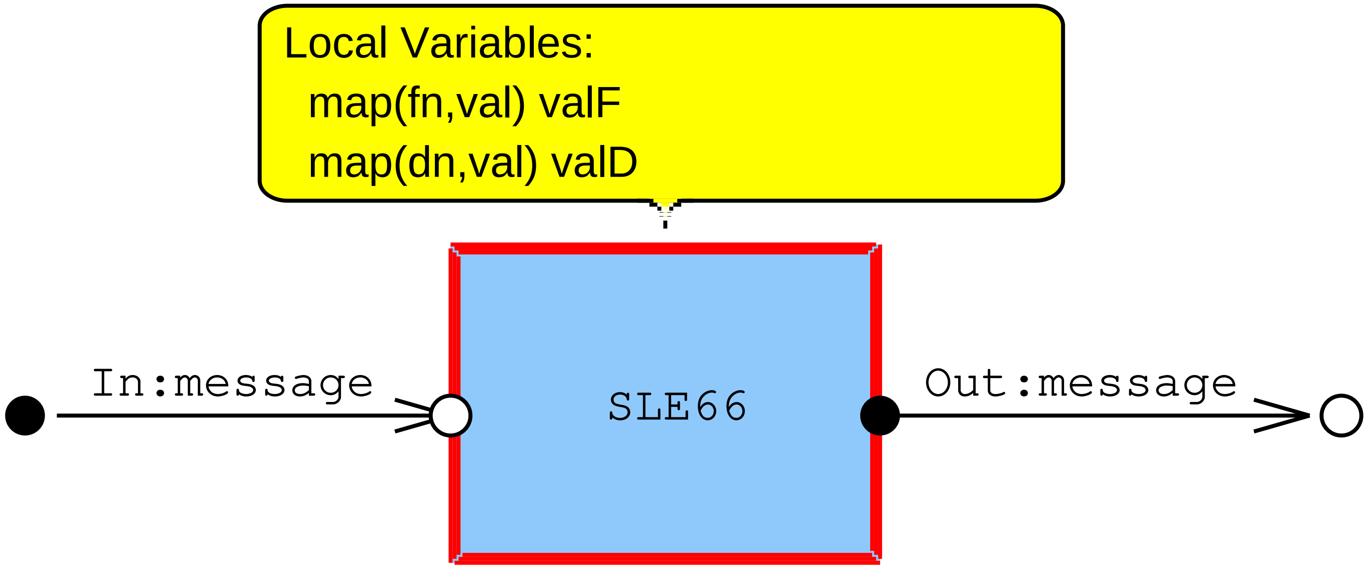
Intrinsic value: Feedback for development and quality control

Abstract system model based on an ad-hoc automaton formalism
Formalization of security requirements, verification

Minor syntactical, typographical and semantical slips
Type errors, missing assumptions, incomplete proofs

$\Rightarrow$ Use of Isabelle/HOL + ISMs

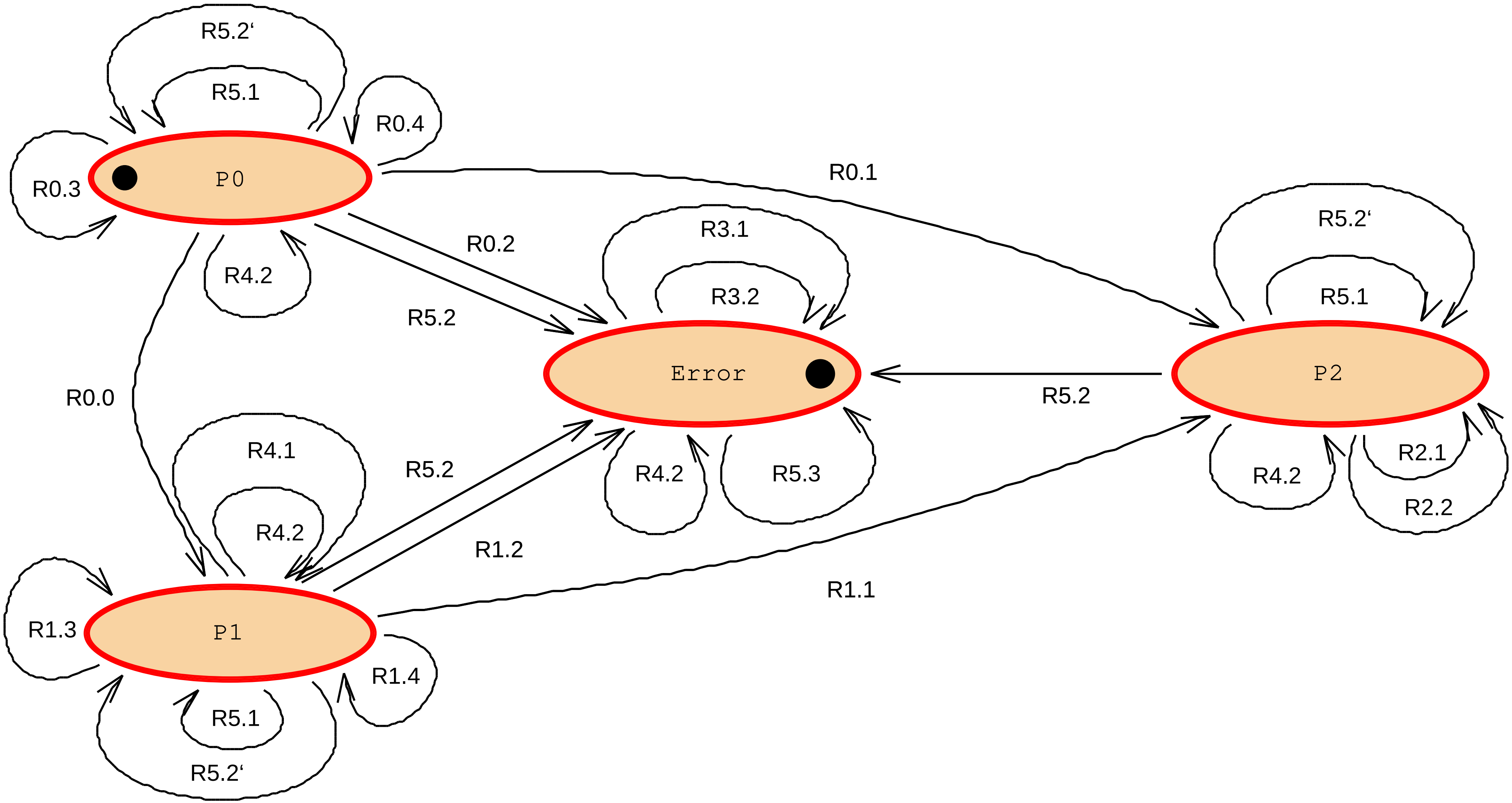
LKW Model: System Architecture



LKW Model: ISM section

```
ism SLE66 =
  ports interface
    inputs   "{In}"
    outputs  "{Out}"
    messages message
  state
    control P0 :: ph
    data    σ0 :: data
  transitions
   .
   .
   .
end
```

LKW Model: State Transitions



LKW Model: Rule 5.2

```
R5.2: ph -> Error
      pre  "ph ≠ Error", "oname ∈ Sec",
           "v ∈ {[], [Val (the (val σ oname))]}"
      in   In  "[Spy oname]"
      out  Out "v"
      eff  valF := fs, valD := ds
```

Typical:

Both input and output

Underspecification

Nondeterminism

Generic transitions

LKW Model: Properties

Abstract specification: ISM + a few axioms, e.g.

$Axiom1$: "$f \in fct\ \sigma \cap F\_Sec \Longrightarrow valF\ (change\ f\ \sigma)\lfloor F\_Sec = valF\ \sigma\lfloor F\_Sec$"

Security objectives: predicates on the system behavior, e.g.

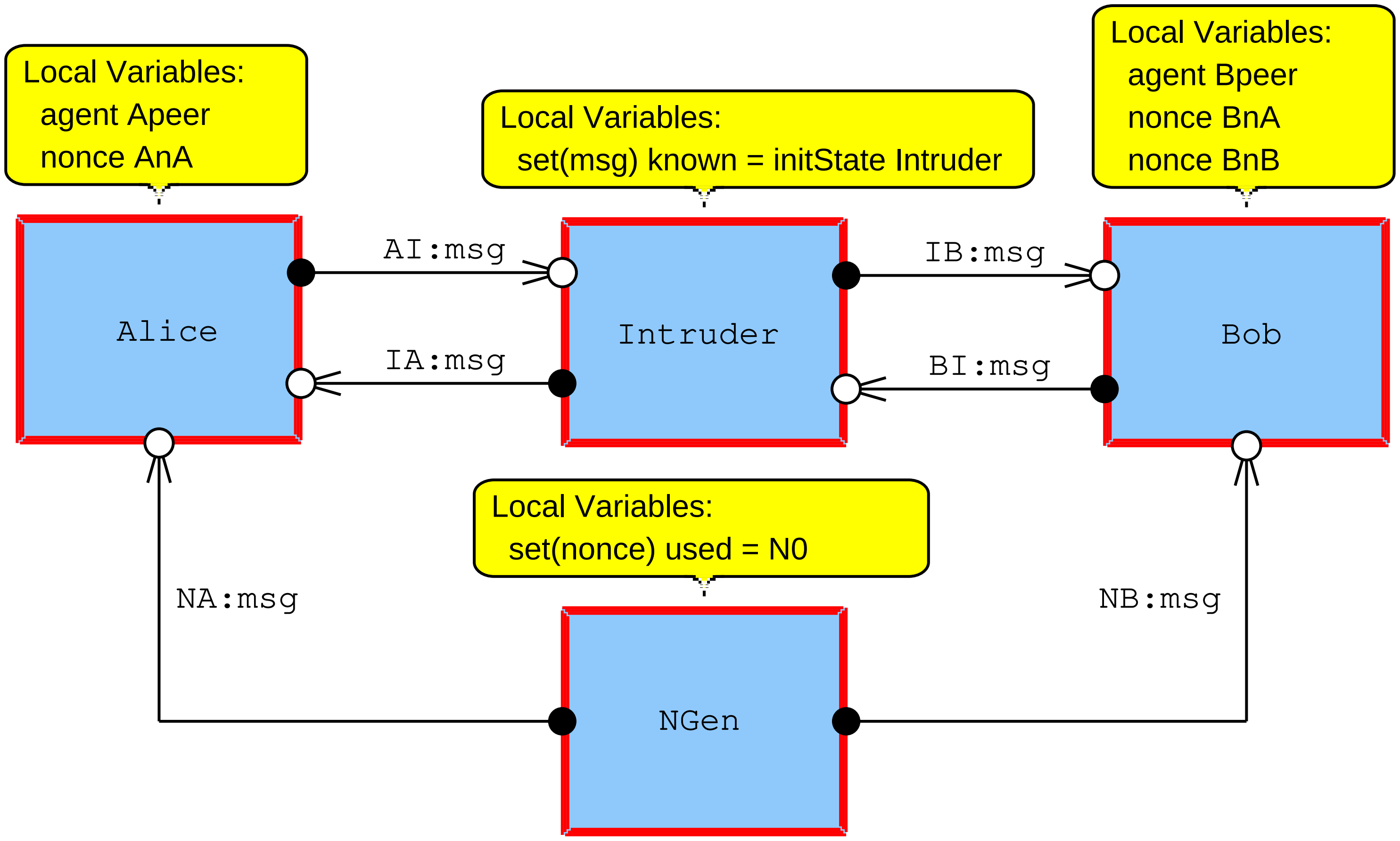
theorem $FS05$: "$[\![((ib,(\_,\sigma)),p,(\_,(\_,\sigma')))\in Trans;\ ib\ In = Exec\ sb\ f\#r;$
$f \in FTest]\!] \Longrightarrow sb = Pmf\ \vee\ p\ Out = [No] \wedge \sigma' = \sigma$"

Experience:

- Detected omissions: one axiom, one invariant
- Isabelle proofs: just a few steps, 50% automatic
- New requirements require only slight changes

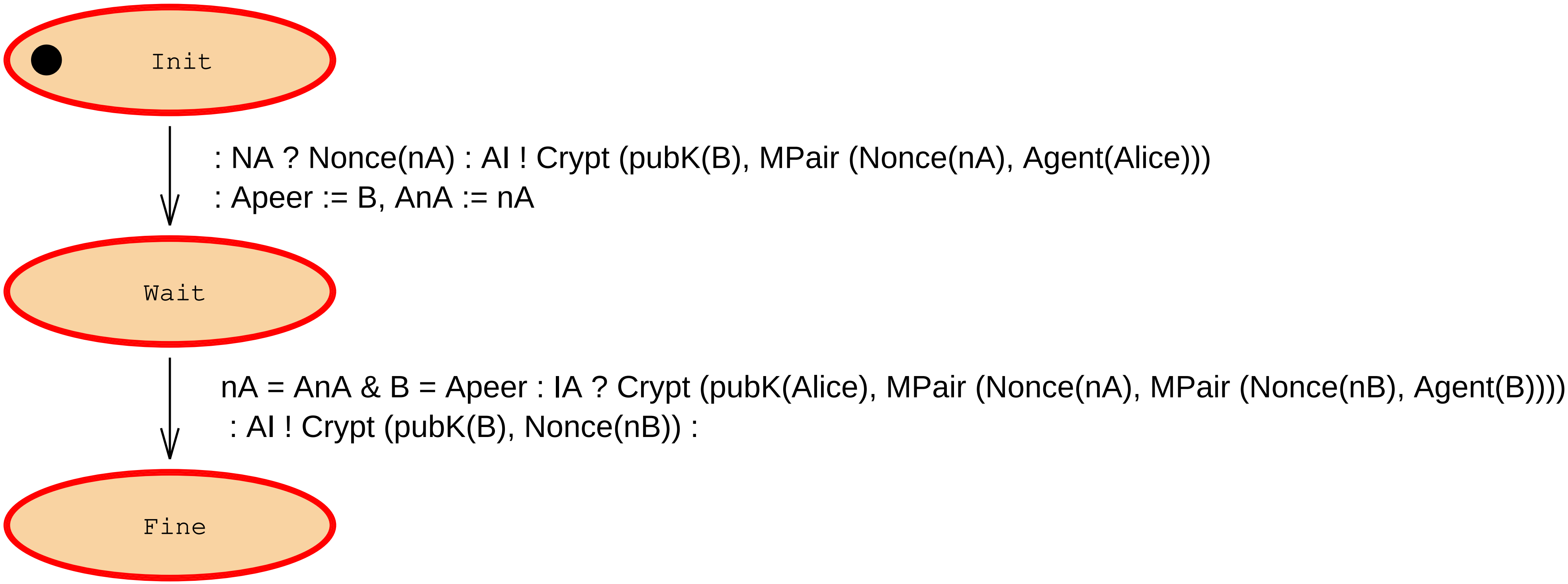
Needham-Schroeder Public-Key Protocol

NSL (Lowe's fix) as benchmark for security in distributed systems



NSL State Transition Diagrams: `Alice`

Transition format: [guard] : [inputs] : [outputs] : [assignments]



Agents do have state: current expectations and knowledge

NSL: Properties

Authentication of `Alice` for Bob (even session agreement!)

Paulson's formulation referring to all messages of the protocol

$$\llbracket A \notin \mathit{bad};\ B \notin \mathit{bad};\ \mathit{evs} \in \mathit{ns\_public};$$
$$\mathit{Crypt}\ (\mathit{pubK}\ B)\ (\mathit{Nonce}\ \mathit{NB}) \in \mathit{parts}\ (\mathit{spies}\ \mathit{evs});$$
$$\mathit{Says}\ B\ A\ (\mathit{Crypt}\ (\mathit{pubK}\ A)\ \{\!|\mathit{Nonce}\ \mathit{NA}, \mathit{Nonce}\ \mathit{NB}, \mathit{Agent}\ B|\!\}) \in \mathit{set}\ \mathit{evs}$$
$$\rrbracket \Longrightarrow$$
$$\mathit{Says}\ A\ B\ (\mathit{Crypt}\ (\mathit{pubK}\ B)\ \{\!|\mathit{Nonce}\ \mathit{NA}, \mathit{Agent}\ A|\!\}) \in \mathit{set}\ \mathit{evs}$$

ISM formulation with reference to agent state

$$\llbracket \mathit{Alice} \notin \mathit{bad};\ \mathit{Bob} \notin \mathit{bad};\ (b,s)\#\mathit{cs} \in \mathit{Runs};$$
$$\mathit{Bob\_state}\ s = (\mathit{Conn},\ (\!|\mathit{Bpeer} = \mathit{Alice},\ \mathit{BnA} = \mathit{nA},\ \mathit{BnB} = \_|\!))\rrbracket \Longrightarrow$$
$$\exists (\_,s') \in \mathit{set}\ \mathit{cs}.$$
$$\mathit{Alice\_state}\ s' = (\mathit{Wait},\ (\!|\mathit{Apeer} = \mathit{Bob},\ \mathit{AnA} = \mathit{nA}|\!))$$

More detail $\Rightarrow$ less automatic, but more insights

Proof using a variant of Schneider's rank function approach

Conclusion

- ISMs support system modeling adequately
- Machine checking reduces errors and omissions
  like hidden assumptions and sloppy argumentation
- Graphical representation suits design and documentation
- ISM framework applicable to a variety of security analysis tasks:
  High-level security modeling and requirements analysis
  Lowel-level analysis of distributed systems like crypto protocols

$\Rightarrow$ ISMs provide good support for practical formal security analysis

Future work: more applications, extended proof support