Interacting State Machines

and their applications in security analysis

NICTA Workshop on Operating System Verification

05th October 2004, Sydney, Australia

David von Oheimb and Volkmar Lotz

Siemens CT IC Sec

{david.von.oheimb|volkmar.lotz}@siemens.com

Overview

- Motivation
- Interacting State Machines (ISMs)
  - Concepts
  - Semantics
  - Tool Support
- Infineon SLE66
- Needham-Schroeder Protocol
- ISM Extensions
  - Dynamic ISMs
  - Ambient ISMs
- Conclusion
- Selected References

Motivation of Formal Analysis

- **Our customers:** IT developers with security concerns
  - Requirements analysis for security, e.g. Siemens Med
  - Evaluation according to ITSEC and CC, e.g. Infineon
- **Our mission:** rigorous security analysis
  - security modeling and verification using formal methods
  - checks and presentation done with machine assistance
- **First challenge:** which framework shall we employ?

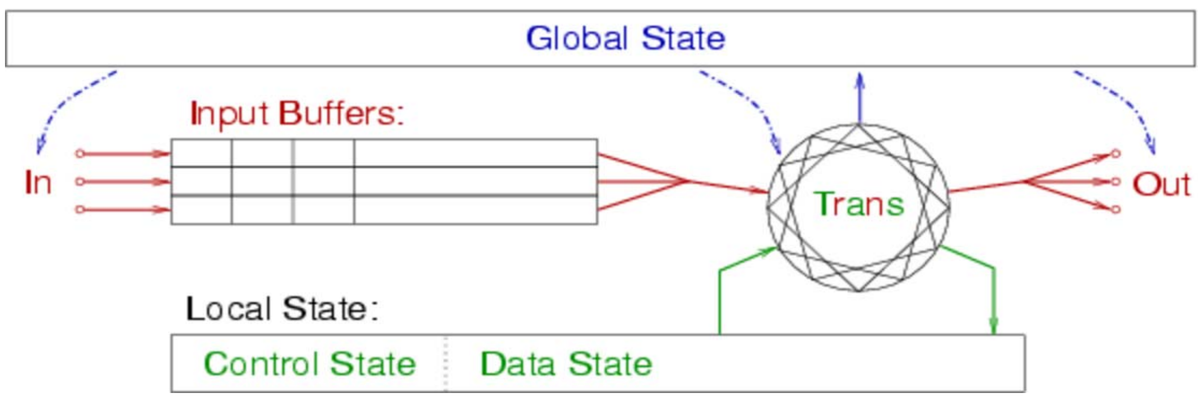
Requirements for Formalism

- **Expressiveness**: state transformation, concurrency, messaging
- **Flexibility**: adaptation and extension
- **Simplicity**: minimal expertise and time
- **Maturity** of the semantics: refinement etc.
- **Graphical** capabilities: overview and intuition
- **Tool support**: mature and freely available

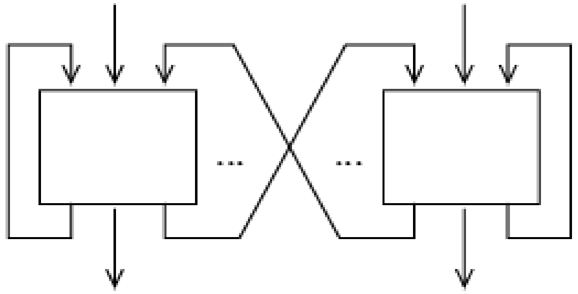
CORPORATE TECHNOLOGY

Interacting State Machines (ISMs)

- state transitions (maybe non-deterministic)
- buffered I/O simultaneously on multiple connections



- finite trace semantics
- modular (hierarchical) parallel composition



Information &
Communications
Security

CORPORATE TECHNOLOGY

Formal Definition of Basic ISMs

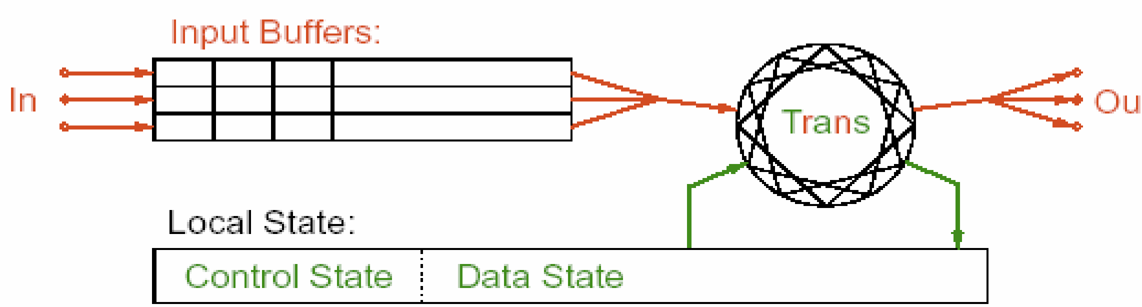
$MSGs = \mathcal{P} \to \mathcal{M}^*$ — family of message sequences $\mathcal{M}$, indexed by port names $\mathcal{P}$

$CONF(\Sigma) = MSGs \times \Sigma$ — configuration with local state Σ

$TRANS(\Sigma) = \wp((MSGs \times \Sigma) \times (MSGs \times \Sigma))$ — transitions

$ISM(\Sigma) = \wp(\mathcal{P}) \times \wp(\mathcal{P}) \times \Sigma \times TRANS(\Sigma)$ — ISM type

$a = (In(a), Out(a), \sigma_0(a), \ Trans(a))$ — ISM value a



Information & Communications Security

Open runs

$$Runs(a) \in \wp(\Sigma^*)$$

$$\langle \sigma_0(a) \rangle \in Runs(a)$$

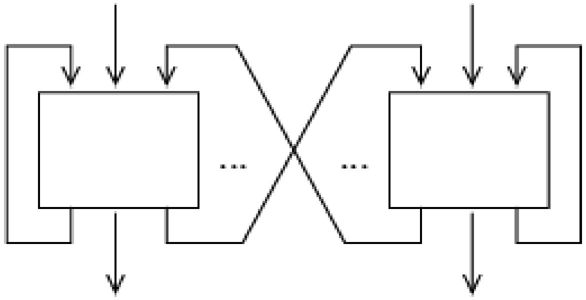
$$\frac{\begin{array}{c} ss^\frown\sigma \in Runs(a) \\ ((i,\sigma),(o,\sigma')) \in Trans(a) \end{array}}{ss^\frown\sigma^\frown\sigma' \in Runs(a)}$$

CORPORATE TECHNOLOGY

Parallel Composition

Let $A = (A_i)_{i \in I}$ be a family of ISMs. Their *parallel composition* $\|_{i \in I} A_i$ is an ISM of type $\mathit{ISM}(\mathit{CONF}(\Pi_{i \in I} \Sigma_i))$ being defined as

$$(\mathit{AllIn}(A) \setminus \mathit{AllOut}(A),\ \mathit{AllOut}(A) \setminus \mathit{AllIn}(A),\ (\varnothing, S_0(A)),\ \mathit{PTrans}(A))$$

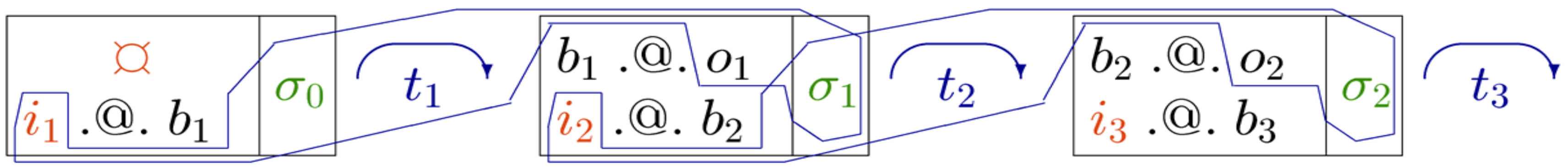


where

- $\mathit{AllIn}(A) = \bigcup_{i \in I} \mathit{In}(A_i)$
- $\mathit{AllOut}(A) = \bigcup_{i \in I} \mathit{Out}(A_i)$
- $S_0(A) = \Pi_{i \in I}(\sigma_0(A_i))$ is the Cartesian product of all initial local states
- $\mathit{PTrans}(A) \in \mathit{TRANS}(\mathit{CONF}(\Pi_{i \in I} \Sigma_i))$ is the parallel composition of their transition relations, defined as . . .

Information & Communications Security

Parallel State Transition Relation



$$\frac{j \in I \qquad ((i,\sigma),(o,\sigma')) \in \mathit{Trans}(A_j)}{\begin{array}{l}((i_{|\overline{AllOut(A)}},\ (i_{|AllOut(A)}\ .@.\ b,\ S[j := \sigma])),\\ \ (o_{|\overline{AllIn(A)}},\ (b\ .@.\ o_{|AllIn(A)},\ S[j := \sigma']))) \in \mathit{PTrans}(A)\end{array}}$$

where

- $S[j := \sigma]$ replaces the j-th component of the tuple S by σ
- $m_{|P}$ denotes the restriction $\lambda p.$ if $p \in P$ then $m(p)$ else $\langle\rangle$ of the message family m to the set of ports P
- $o_{|\overline{AllIn(A)}}$ denotes those parts of the output o provided to any outer ISM
- $o_{|AllIn(A)}$ denotes the internal output to peer ISMs or direct feedback, which is added to the current buffer contents b

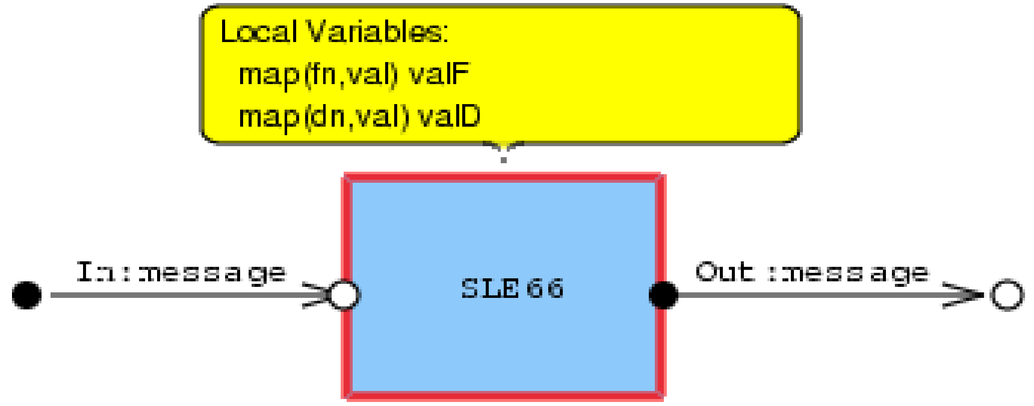
Tool Support

- **AutoFocus: CASE tool for graphical specification and simulation**
  - syntactic perspective
  - graphical documentation
  - type and consistency checks
- **Isabelle/HOL: powerful interactive theorem prover**
  - semantic perspective
  - textual documentation
  - validation and correctness proofs
- AutoFocus drawing → Isabelle theory file

  Within Isabelle: ism sections → standard HOL definitions

Graphical Representation in AutoFocus:

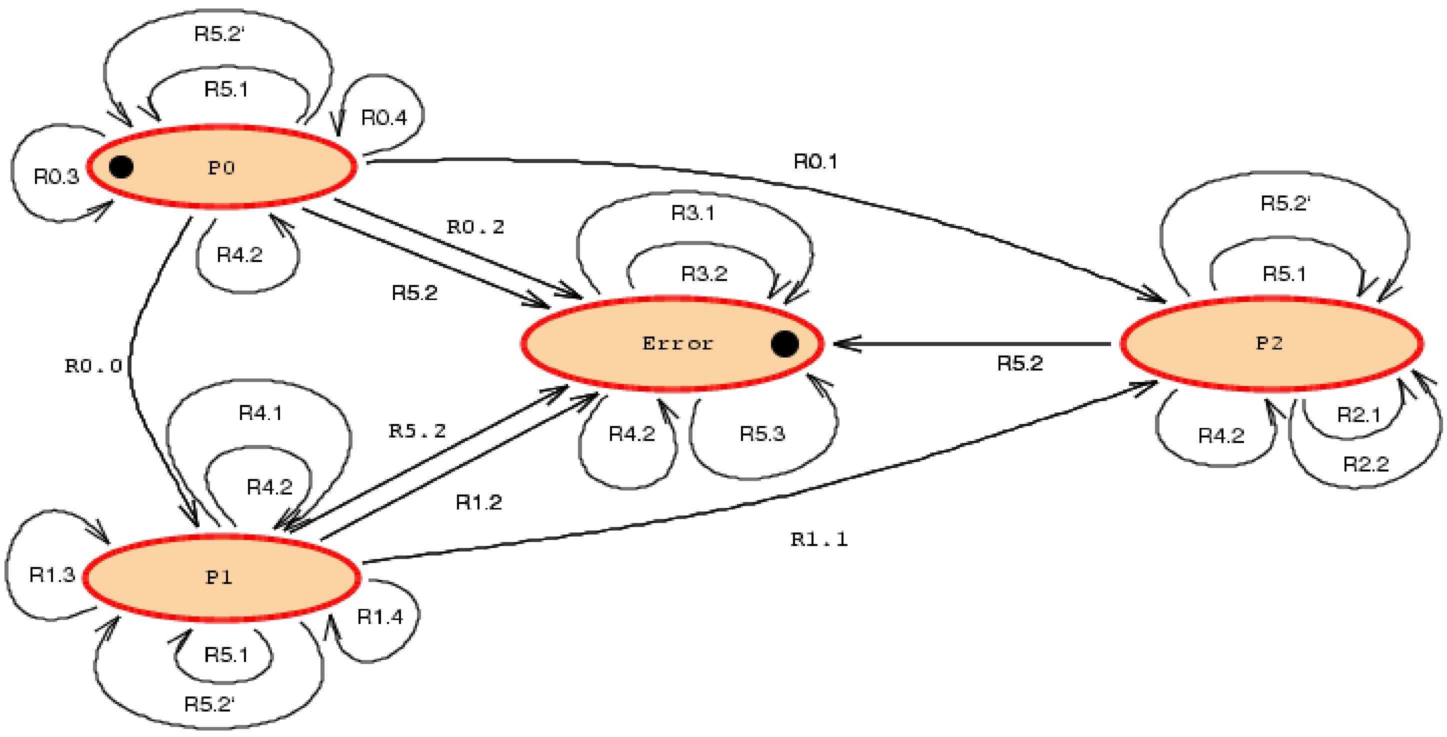
SLE66 System Structure Diagram



CORPORATE TECHNOLOGY

Graphical Representation in AutoFocus:

SLE66 State Transition Diagram



Basic ISMs in Isabelle/HOL

```
ism name ((param_name :: param_type))* =
    ports pn_type
      inputs    I_pns
      outputs   O_pns
    messages msg_type
     states     [state_type]
      [control cs_type [init cs_expr0]]
      [data      ds_type [init ds_expr0] [name ds_name]]
    [transitions
      (tr_name [attrs]: [cs_expr -> cs_expr']
      [pre   (bool_expr)+ ]
      [in    ([multi] I_pn  I_msgs)+ ]
      [out   ([multi] O_pn O_msgs)+ ]
      [post  ((lvar_name := expr)+ | ds_expr')] )+ ]
```

SLE66 ISM section: static part

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

SLE66 ISM section: Transition 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"
      post valF := fs, valD := ds
```

Typical:

- Both input and output in each transition
- Underspecification
- Nondeterminism
- Genericity

CORPORATE TECHNOLOGY

SLE66 model: Properties

- **Abstract specification:** ISM section plus a few axioms, e.g.:

  “security-relevant functions do not modify security-relevant functions”

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

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

  “only the processor manufacturer can successfully call test functions”

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

Experience:

- Detected omissions: one axiom, one invariant
- Proofs in Isabelle: just a few steps, 50% automatic
- New requirements lead to slight changes only

Information & Communications Security

Needham-Schroeder Public-Key Protocol

- **Simple authentication protocol** as defined in 1978

$$\begin{array}{llcl} \text{M1.} & A \to B & : & \{n_A, A\}_{K_B^+} \\ \text{M2.} & B \to A & : & \{n_A, n_B\}_{K_A^+} \\ \text{M3.} & A \to B & : & \{n_B\}_{K_B^+} \end{array}$$

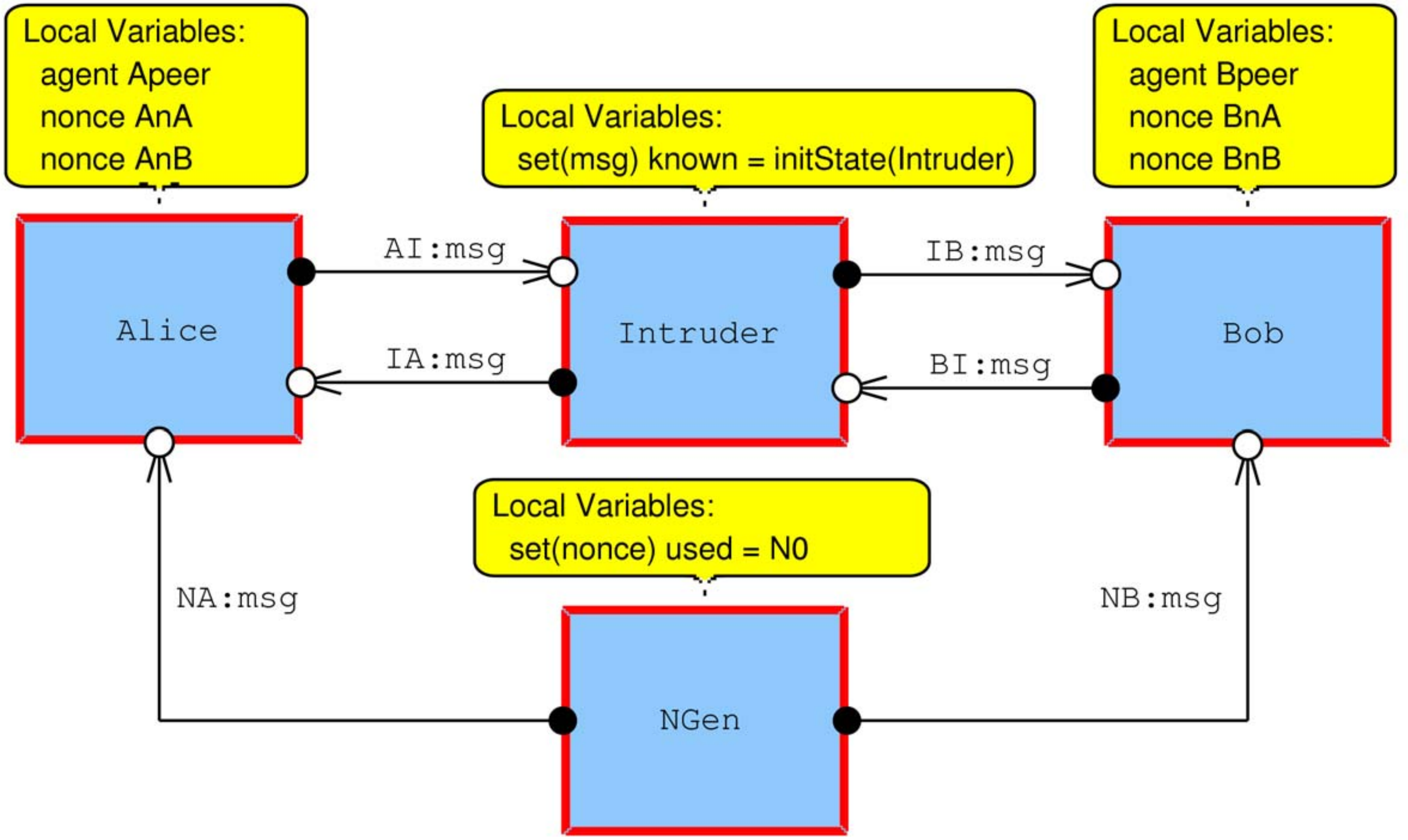
- **Man-in-the-middle attack** found and fixed by Lowe in 1995

$$\begin{array}{lrcl} \text{M1(1).} & A \to I & : & \{n_A, A\}_{K_I^+} \\ \text{M1(2).} & I(A) \to B & : & \{n_A, A\}_{K_B^+} \\ \text{M2(2).} & B \to I(A) & : & \{n_A, n_B\}_{K_A^+} \\ \text{M2(1).} & I \to A & : & \{n_A, n_B\}_{K_A^+} \\ \text{M3(1).} & A \to I & : & \{n_B\}_{K_I^+} \\ \text{M3(2).} & I(A) \to B & : & \{n_B\}_{K_B^+} \end{array}$$

- NSL used as example of simple **security-critical distributed system**

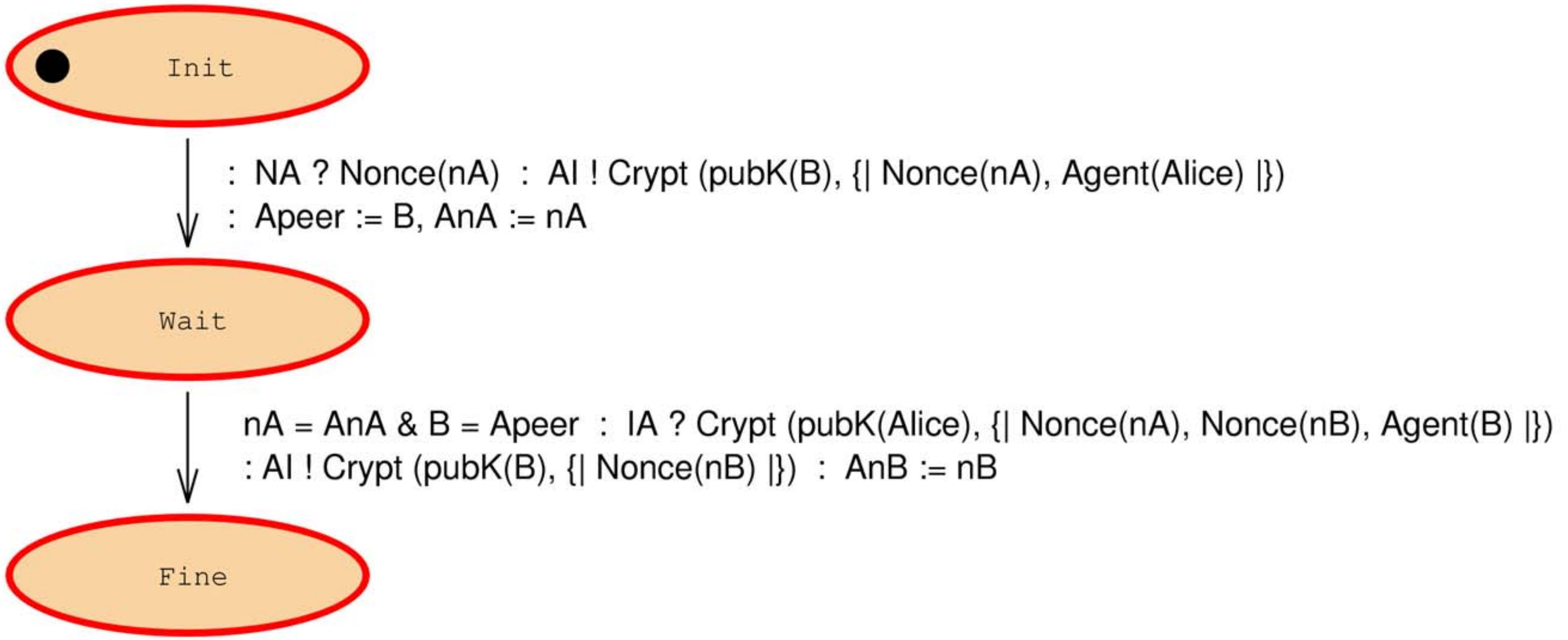
NSL System Structure Diagram

- **Agents** Alice and Bob, Dolev-Yao-style **intruder**, **nonce generator**



NSL State Transition Diagrams

- **Alice** initiates exchange, awaits and acknowledges correct response



Agents do have state: current knowledge and expectations

CORPORATE TECHNOLOGY

NSL Properties

Example: authentication of Alice for Bob (even session agreement)

- **Paulson's formulation** can refer only to messages sent

$$
\begin{array}{l}
[\![A \notin bad;\ B \notin bad;\ evs \in ns\_public; \\
\quad Crypt\ (pubK\ B)\ (Nonce\ NB) \in parts\ (spies\ evs); \\
\quad Says\ B\ A\ (Crypt\ (pubK\ A)\ \{\!|Nonce\ NA, Nonce\ NB, Agent\ B|\!\}) \in set\ evs \\
]\!] \Longrightarrow \\
\quad Says\ A\ B\ (Crypt\ (pubK\ B)\ \{\!|Nonce\ NA, Agent\ A|\!\}) \in set\ evs
\end{array}
$$

- **ISM formulation** with reference also to agent state

$$
\begin{array}{l}
[\![Alice \notin bad;\ Bob \notin bad;\ (b,s)\#cs \in Runs; \\
\quad Bob\_state\ s = (Conn,\ (\!|Bpeer = Alice,\ BnA = nA,\ BnB = \_|\!))]\!] \Longrightarrow \\
\exists\,(\_,s') \in set\ cs. \\
\quad Alice\_state\ s' = (Wait,\ (\!|Apeer = Bob,\ AnA = nA|\!))
\end{array}
$$

Proofs: more detail → less automatic, but more insights

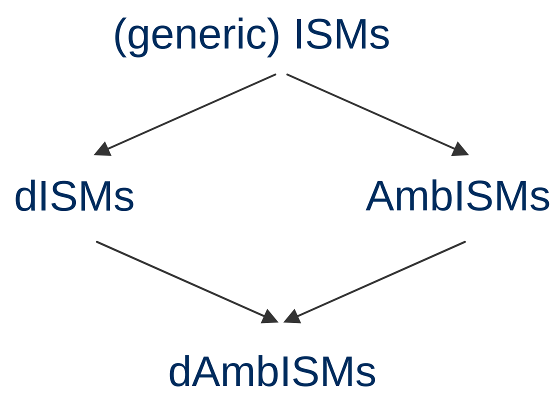
using a variant of Schneider's rank function approach

Information &
Communications
Security

CORPORATE TECHNOLOGY

Extensions to ISM Concepts

- **Generic ISMs:** global/shared state
- **Dynamic ISMs:** changing availability and connection patterns
- **Ambient ISMs:** mobility with constrained communication
- **Dynamic Ambient ISMs:** combination



- **Application:** German BMWA lead project MAP

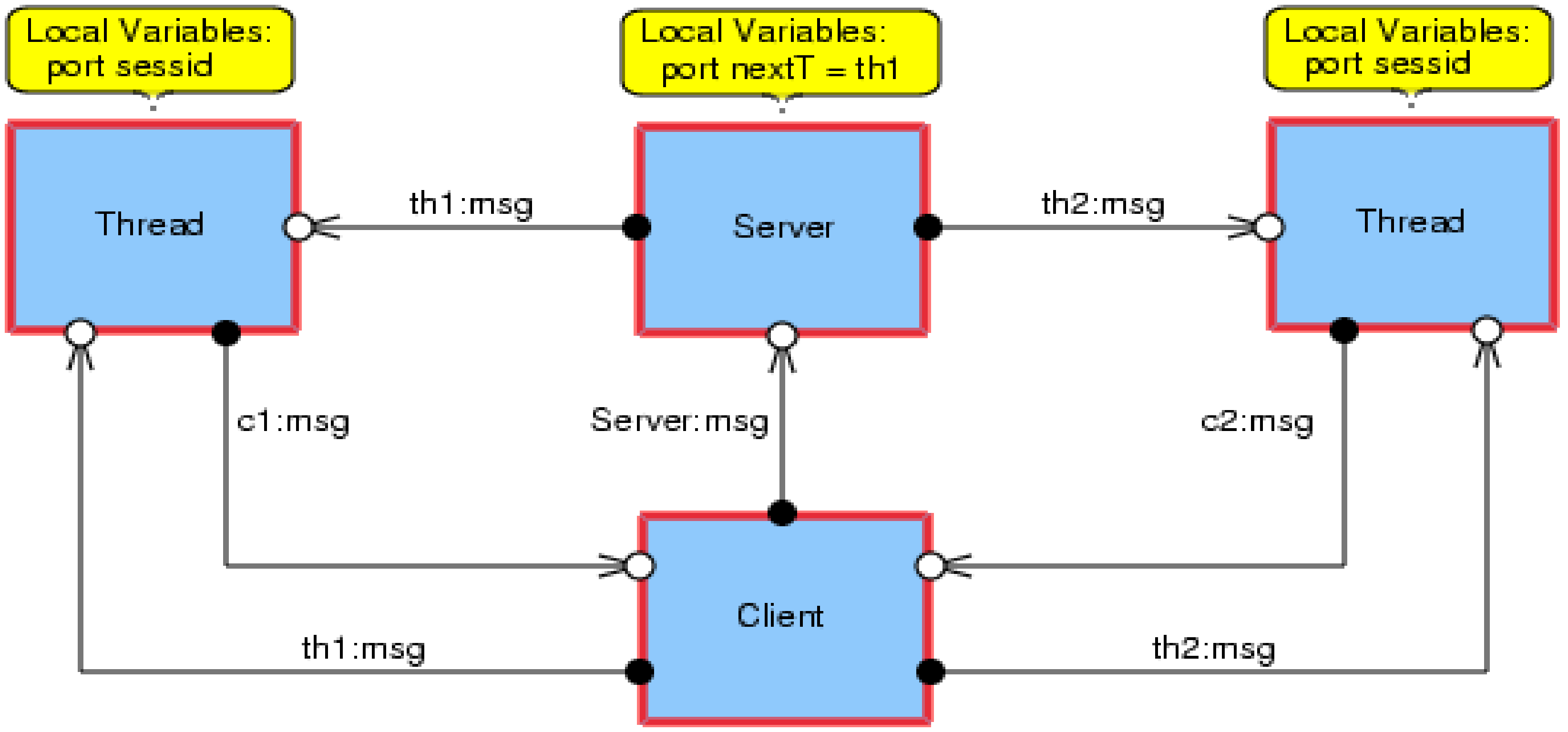
  "Mobile workplace of the future" (Thomas Kuhn)

Information & Communications Security

CORPORATE TECHNOLOGY

Dynamic ISM example

- **System Structure Diagram:** client/server (multithreaded)

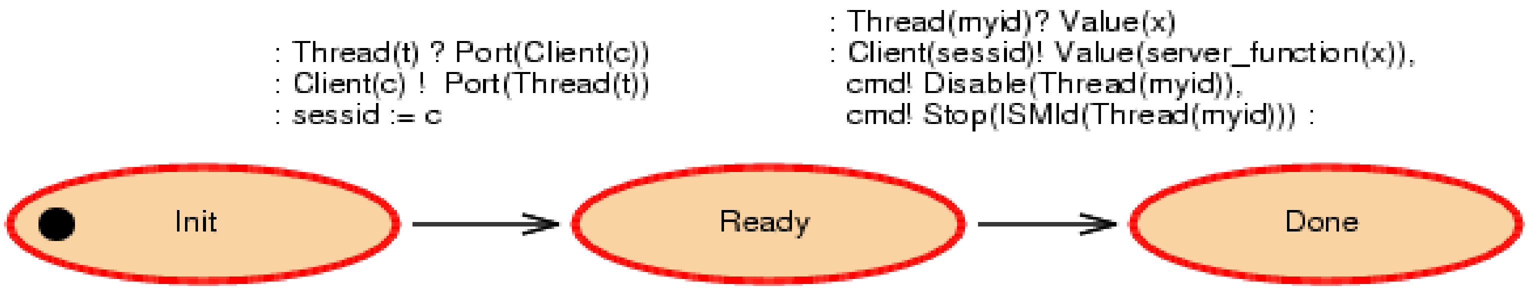


For each client request, the server activates a new worker thread, creates a new port and conveys it to the new thread.

Information & Communications Security

Dynamic ISM example

- **State Transition Diagram:** worker thread



The thread receives the client port, sends its own port to the client, receives a value, transforms it, and sends it back to the client. Finally, the thread disables its port and stops.

CORPORATE TECHNOLOGY

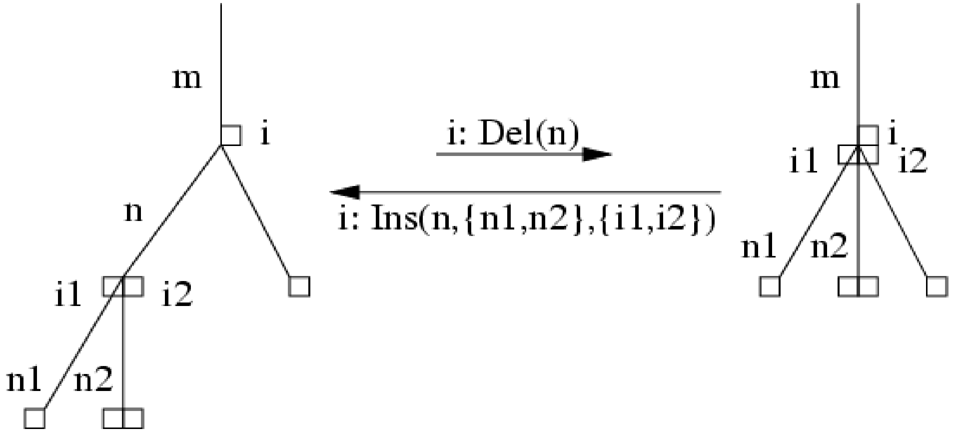
Dynamic Ambient ISMs

- Dynamic commands:
  Run(i), Stop(i), Enable(p), Disable(p), New(p), Convey(p,i)
- Additional structure:

  ambient tree

  governing locality

  constraints



- Mobility commands:
  Assign(i,n), In(n), Out(n), Del(n), Ins(n,ns,is)
- Operational semantics of Ambient Calculus

Information & Communications Security

Ambient ISM Example

- **Homebase places an agent in its environment**

```
start:
 Start -> Instruct
 cmd “[Ins AG_amb {} {}, Assign AG AG_amb]“
```

- **Agent gets the route imprinted**

```
 out AGData “[Route [HB_amb, AP_amb 1, AP_amb 2,
                                     HB_amb]]“
```

- **Agent migrates to the next agent platform on the route**

```
migrate:
 Migrate -> Decide
 pre “route s = r#rs“
 cmd “[Out (here s), In r]“
 post here := r, route := rs
```

Our Applications of ISMs

- Infineon SLE 66 smart card processor [LKW]
- Infineon SLE 88 memory management [OWL]
- mobile agent case study for MAP project [KO]
- access control for medical information system
- document management system for aviation industry

Conclusion

- ISMs allows to **model** systems **adequately**
- **Graphical representation** suits design and documentation
- **Machine checking** reduces errors and omissions
  like hidden assumptions and sloppy argumentation

- ISM framework **applicable to a variety** of security analysis tasks
  - **High-level** security modeling and requirements analysis
  - **Low-level** analysis of distributed systems like crypto protocols

→ **ISMs provide good support for practical formal security analysis**

- **Future work:** test case generation, refinement, …

Selected References

- D. v.Oheimb, V. Lotz,
  "Formal Security Analysis with Interacting State Machines", ESORICS 2002

- D. v.Oheimb, V. Lotz,
  "Extending Interacting State Machines with Dynamic Features", ICFEM 2003

- T. Kuhn, D. v.Oheimb,
  "Interacting State Machines for Mobility", FM 2003

- D. v.Oheimb, G. Walter, V. Lotz,
  "A Formal Security Model for the Infineon SLE88 Smartcard Memory Management", ESORICS 2003

Backup Slides

- Parallel ISM runs
- Isabelle/HOL
- Project MAP

Parallel Runs (with Interaction)

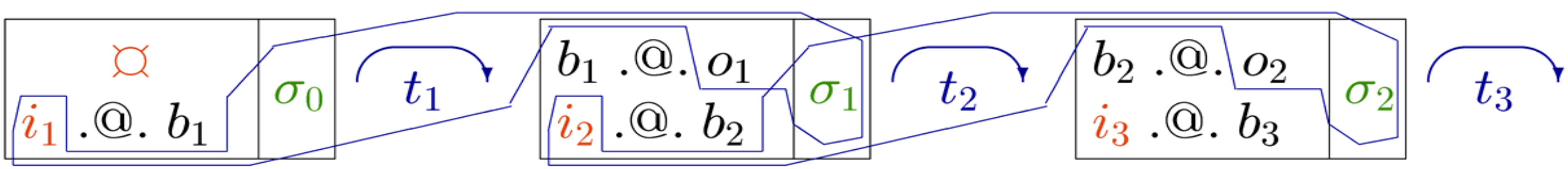
Let $A = (A_i)_{i \in I}$ be a family of ISMs.

$CRuns(A)$ of type $\wp((CONF(\Pi_{i \in I} \Sigma_i))^*)$

$$\frac{}{\langle(\text{¤}, \Pi_{i \in I}(\sigma_0(A_i)))\rangle \in CRuns(A)}$$

$$\frac{\begin{array}{c} j \in I \\ cs ^\frown (i\ .@.\ b, (S[j := \sigma])) \in CRuns(A) \\ ((i, \sigma), (o, \sigma')) \in Trans(A_j) \end{array}}{cs ^\frown (i\ .@.\ b, S[j := \sigma]) ^\frown (b\ .@.\ o, S[j := \sigma']) \in CRuns(A)}$$

$S[j := \sigma]$ replaces the j-th component of the tuple S by σ.



Isabelle/HOL

- generic interactive theorem prover
- most popular object logic: Higher-Order Logic (HOL)
  (for its expressiveness + automatic type inference)
- HOL: predicate logic based on simply-typed lambda-calculus
- proofs with semi-automatic tactics including rewriting
- user interface: Proof General, integrated with XEmacs
- well-documented and supported, freely available (open-source)

Project MAP

- **MAP: „Multimedia Arbeitsplatz der Zukunft“**
- **One of the six main projects in the area of**
  Integrating Man and Machine in the Knowledge Society
  sponsored by the German Federal Ministry of Economics and Labor
- **Partners: Industrial (9), SME (5), Academic (6)**
- **Aim: develop novel concepts and a basis for future mobile,**
  multi-media based work places
- **Methods from**
  - security technology
  - man-machine interaction
  - agent technology
  - Mobility support