Information Flow Control to Ensure the Security of Multiagent Systems

Dieter Hutter

(joined work with H. Mantel, I.Schaefer, A.Schairer)

German Research Center for Artificial Intelligence

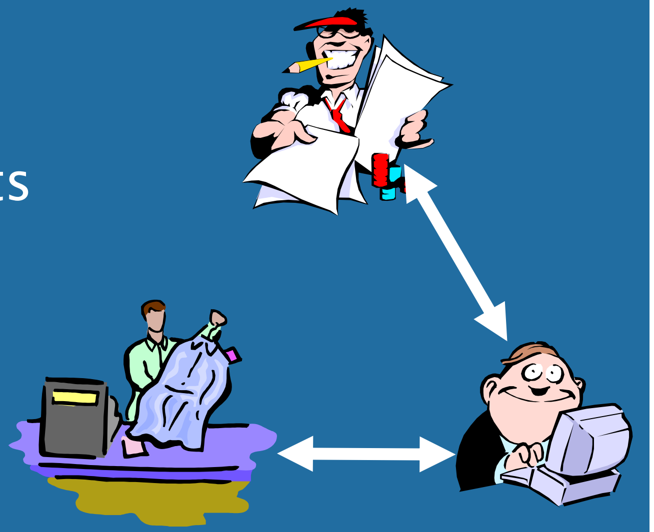
Saarbrücken

Secure Agents: Comparison Shopping

Formal Specifications of Comparison Shopping

Basic scenario:

- Customer agents are initialized with preferences
- Customers request offers from all merchants
- Customers store offers until timeout
- Customers select best offer according to intial preferences

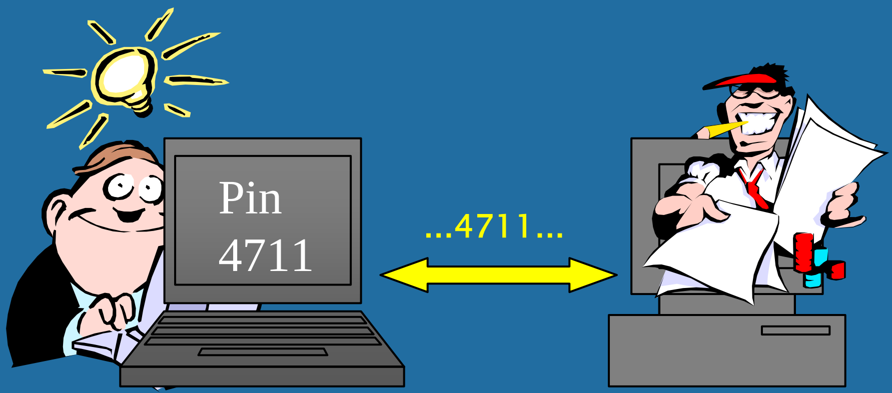


Variations of basic scenario:

- Customer buys the best offer
- Customer considers budget
- Negotiations about price
- Anonymous collection of offers by customers

Confidentiality as Information Flow Control

Confidentiality as a
property of dependency



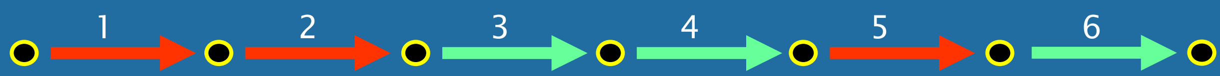
- Changing a secret (e.g. 4711) will not affect visible behaviour (e.g. 4711 on transmission)
- Specification using possibilistic information flow policy

Development of a framework MAKS to specify and verify possibilistic information flow

System Model – Information Flow Control

- Observer has complete knowledge on system behaviour
- Visible events must not depend on secret events
- i.e. set of possible traces (system runs) has to include a trace in which the secret event did not happen. (closure property)

Trace-based system model



Event system: ES = (E, I, O, Tr)

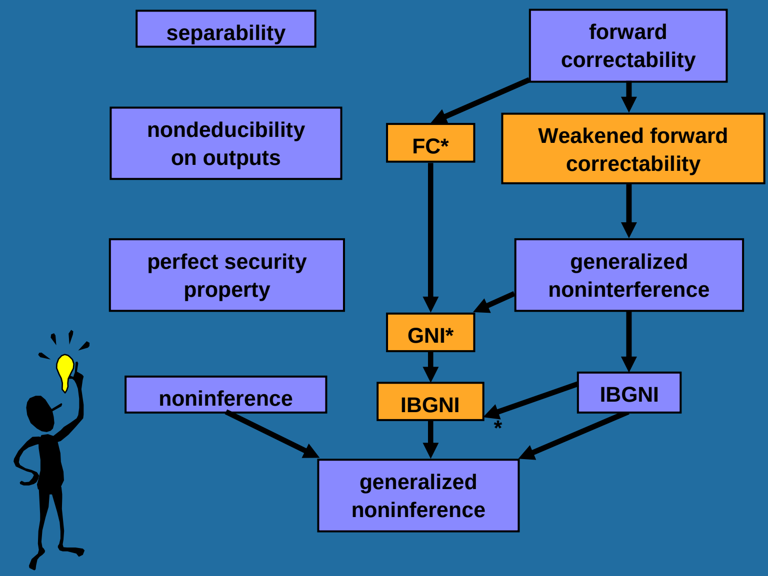
- E set of events, e.g. *set-bal(5342), send(licence,4711)*
- I,O ⊆ E Input/Output events
- Tr ⊆ E* set of admissible traces (prefix closed)

MAKS – Information Flow Control

- View (V, N, C)
  - Events are split into confidential (C), visible (V) and non-visible (N) (but not confidential) events
    (views are local to individual observers)

- Basic security predicates
  - Properties on sets of traces (system behaviour!) wrt. a view
  - Closure properties (!)

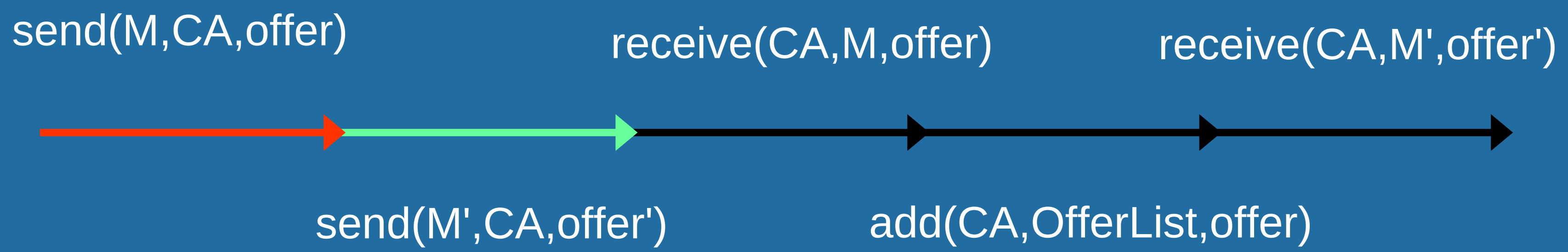
- Security predicates
  - Conjunction of basic security predicates



Formalization of Global Security Requirements

Definition of visible, non-visible and confidential events for all individual observer M‘ :

Confidential: offer of M to CA

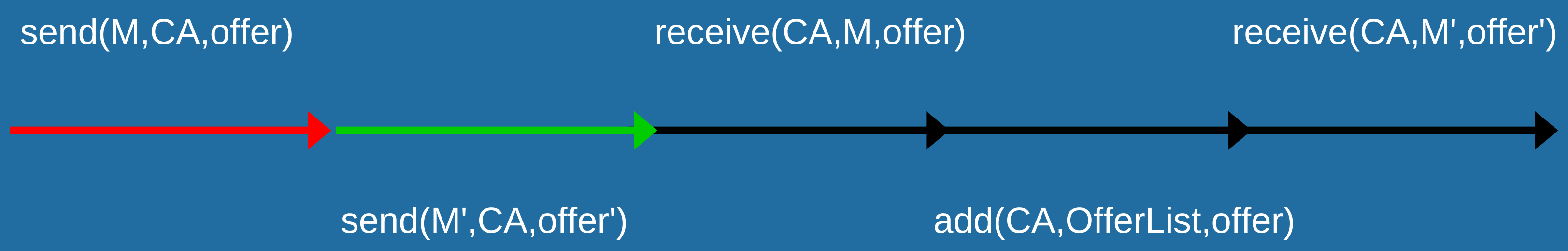


Formal: definition of a view $\mathcal{V} = (V,N,C)$ as partition of the set of events

Formalizing Confidentiality

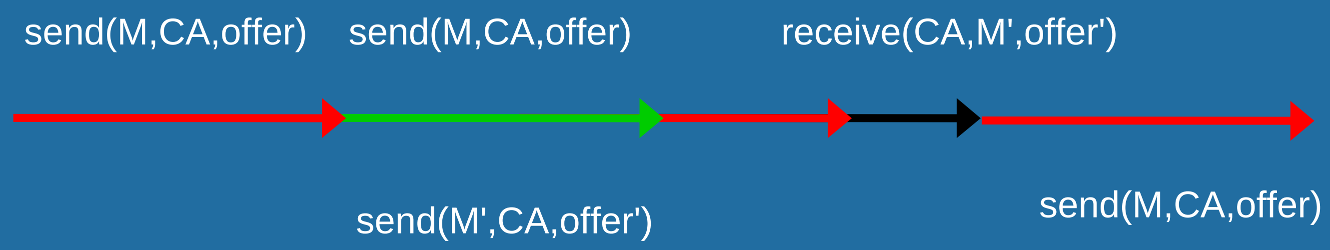
1. Keeping the occurrence of an event confidential (view of M')

BSD (backward strict deletion)



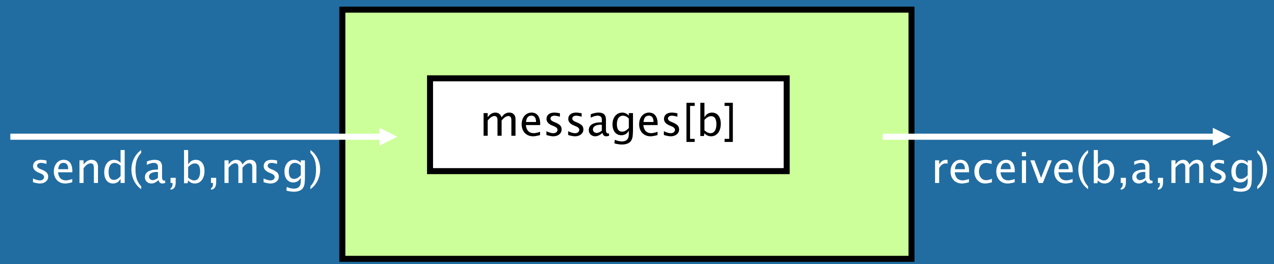
2. Keeping the non-occurrence of events confidential

BSIA (backwards strict insertion of admissible events)

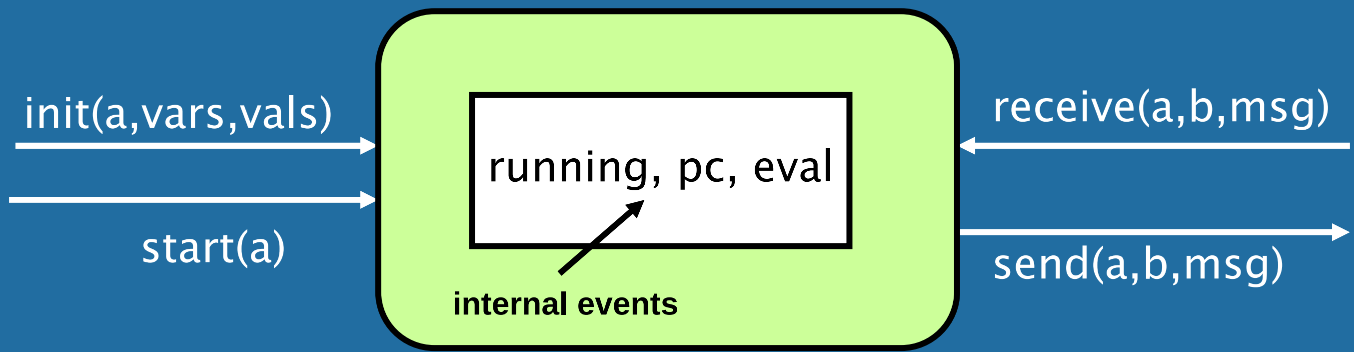


Formalizing Platform and Agents

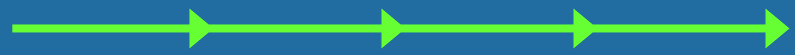
Platform State-Event System $SES_P = (E_P, I_P, O_P, S_P, s^0{}_P, T_P)$

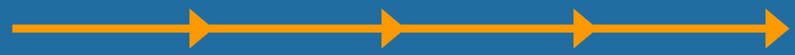


Agent State-Event System $SES_a = (E_a, I_a, O_a, S_a, s^0{}_a, T_a)$



Composition of Agents and Platforms

Agent 1 

Agent 2 

Composition of agent 1 and agent 2:

Example:

Formal Modeling of System Architecture

Admissible traces of componed system:

- Interleaving the traces of the components
- Synchronization by shared events
- Output events of one component can be input event of another component
- Using composition results for basic security predicates

Definition: ES_1 and ES_2 are composable iff

- $E_1 \cap E_2 \subseteq (I_1 \cap O_2) \cup (I_2 \cap O_1)$

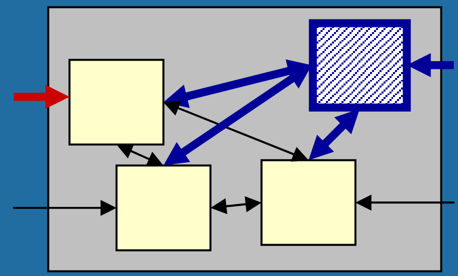
Definition: $(E,I,O,Tr) = ES_1 || ES_2$

- $E = E_1 \cup E_2$
- $I = (I_1 \setminus O_2) \cup (I_2 \setminus O_1)$
- $O = (O_1 \setminus I_2) \cup (O_2 \setminus I_1)$
- $Tr = \{ \tau \in E^* \mid T_1|E_1 \in Tr_1 \wedge T_2|E_2 \in Tr_2 \}$

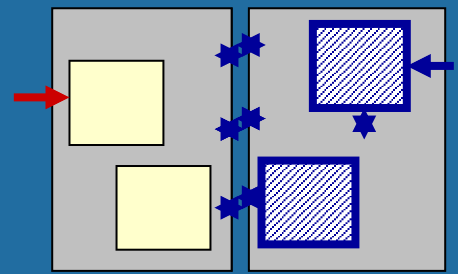
	R	BSD	BSI	BSIA	FCD	FCI	cond
R	X						
BSD		X					
BSI		X	X				
BSIA		X		X			A
FCD		X			X		B,C
FCI		X	X			X	B,D,E

Decomposition of Security Requirements

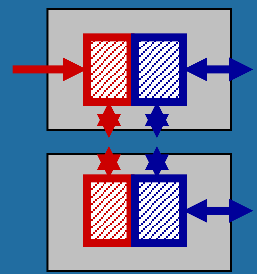
- Global requirement (MAS)
  -> local requirement (Agent)
- Friends (control, trust) versus observers



- Friends' requirements related by side-conditions

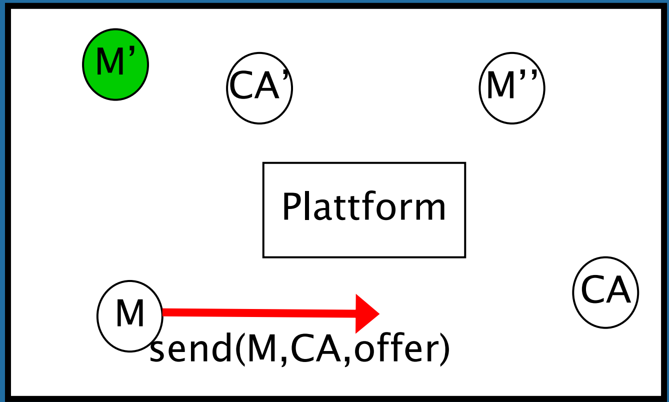


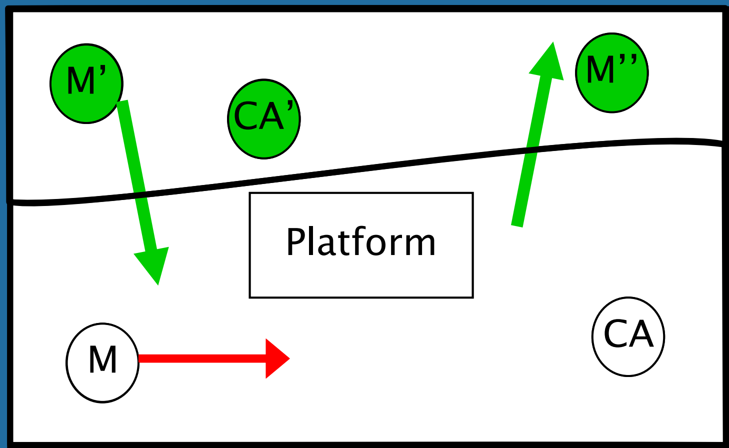
- Completely formal argument
- Read backwards: composition theorem



Strengthening the Proof Conditions

Instead of observer M' consider a coalition of observers





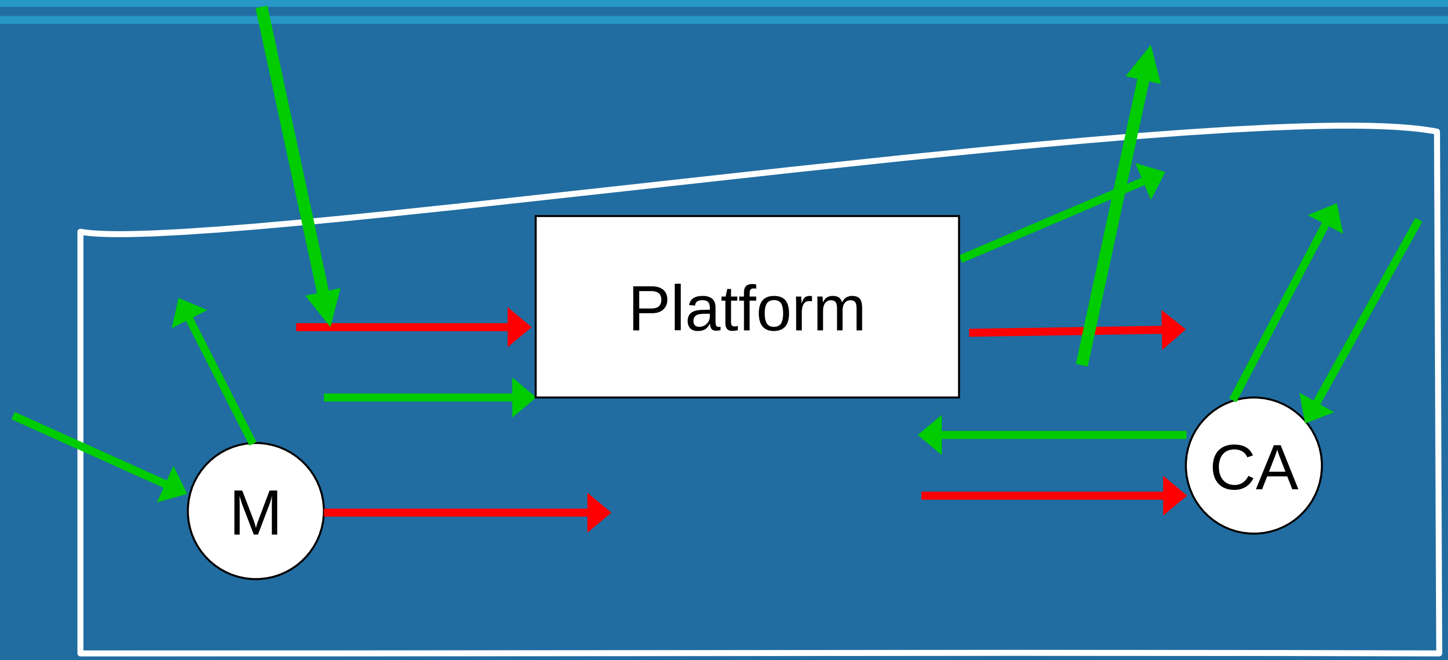
Partitioning in friends and observers

Proof Techniques

- Verification of local security properties of all individual agents
- Composition of individual properties to obtain the overall security properties of the system

- General procedure:
  - Strengthening global security properties
  - Formulating properties for friends and observers
  - Formulating properties for friends
  - Verification of local properties

Local Properties of Friends



- Shared events with observers are visible
- I/O-events are either confidential or visible
- Internal events are non-visible (but not confidential)

Comparison Shopping (including buy event)

Example:

send(M,CA,offer) → send(M',CA,offer') → send(M'',CA,offer'') → [...] → receive(M',CA,buy(offer'))

1. View:
   Observer M'
   confidential offers of M to CA

2. Closure properties:

   BSD: removal of confidential offer ✓

   BSIA: insertion of confidential offer
   - worse offer ✓
   - better offer ✗

Comparison Shopping Including Budget

Example

$\xrightarrow{\text{set(CA,Budget,b)}}$ [...] $\xrightarrow{\text{send(M,CA,offer)}}$ $\xrightarrow{\text{send(M‘,CA,offer‘)}}$ [...] $\xrightarrow{\text{receive(M‘,CA,buy(offer‘))}}$

1. View:

   Observer M’

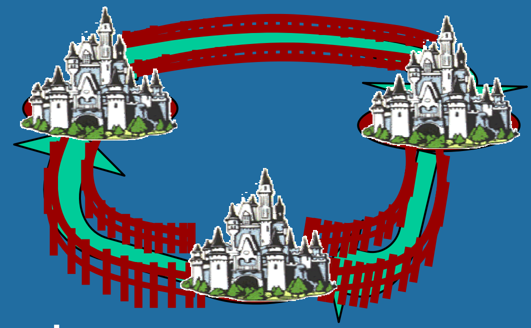
   confidential budget of CA

2. Closure properties:

   BSD: removal of the budget

   BSIA: insertion of a new budget
   - higher budget
   - lower budget

Verification of Comparison Shopping Scenarios



Some results :

- Offers are confidential before purchase
- Purchase reveals merchant that his offer was the best
- Budget remains confidential under specific conditions
- Negotiations can be done independently without interference of offers
- Anonymity can be achieved by extensions of the platform
- ...