Logical Method for Reasoning about Access Control and Data Flow Control Models

Luigi Logrippo
Laboratoire de recherche en sécurité informatique
LRSI
Université du Québec en Outaouais
Gatineau, Québec, Canada

Paper presentation at

7th International Symposium on Foundations and Practice of Security
Montréal, November 2014

Security invariants

- Many security properties can be expressed as *invariant properties of systems*
  - E.g. information of certain types remains within certain boundaries
- However invariants are rarely mentioned and security models are usually defined in terms of *operations* which induce *transformations*

Invariant concept

- In Mathematics a property is *invariant* for certain transformations if *it remains true when these transformations are applied*
  - Concept developed in Computer Science by Floyd, Hoare, Dijkstra, many others

Invariant concept

- In Mathematics a property is *invariant* for certain transformations if *it remains true when these transformations are applied*
  - Concept developed in Computer Science by Floyd, Hoare, Dijkstra, many others
- In Computer Science,
  - the invariant of a program tells what the program is supposed to achieve
  - the program itself tells how this works

Classical Example: Bell La Padula

- Usually described in terms of *transformations* such as:

Subjects cannot
read *information from higher security levels*
nor* write *information to lower ones

- While its *invariant property* could be expressed as:

Information belonging to a security level* can be known *only to subjects of that level or higher

We show that this property remains invariant if the read and write transformations satisfy the conditions specified just above

Isn’t it the same thing?

- *Invariants* make explicit system properties that may not be obvious by looking at the *transformations*
- These are two different views that must agree
  - The one using programming terminology **read**, **write** could be thought of as the *implementation*
  - While the one using the concept of ‘knowledge’ could be thought of as the *specification*
- It must be possible to prove that the implementation corresponds to the specification and vice-versa

Access control and flow control

- Read, write are *access control* concepts
  - Direct relationship between a subject and an object
- Knowledge is a *flow control* concept
  - Where protected values can end up

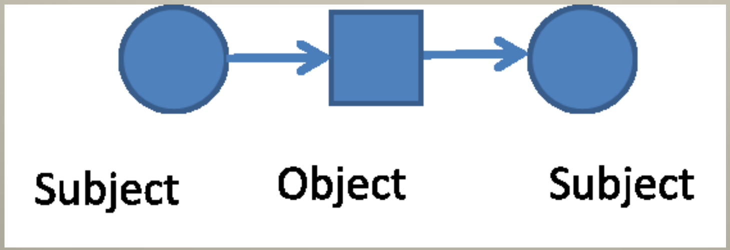
Confidentiality and Integrity invariants

- Confidentiality: information can only be known by authorized subjects
- Integrity: information can only be placed on authorized objects

    - [Sandhu 1993]

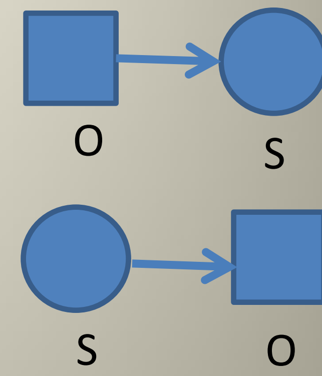
How does information flow?

- In access control systems, information can be *written* by *subjects* on *objects*
- It can be *read* from objects by *subjects*

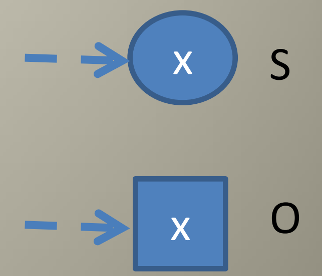


Basic Concepts

- Access Control:
  - CanRead (S,O) : subject S can read from object O
  - CanWrite (S,O) : subject S can write on object O
    - Abbreviated CR, CW



- Flow control:
  - CanKnow (S,x) : subject S can know variable x
  - CanStore (O,x) : object O can contain variable x
    - Abbreviated CK, CS



Flow control *inference rules*

1) *Unconditional relationships* are expressed in the form: CK(S,x) or CS(O,x)

2) *Inference rule* for CK:

$$\exists O\ (CS(O,x) \wedge CR(S,O)) \Rightarrow CK(S,x)$$

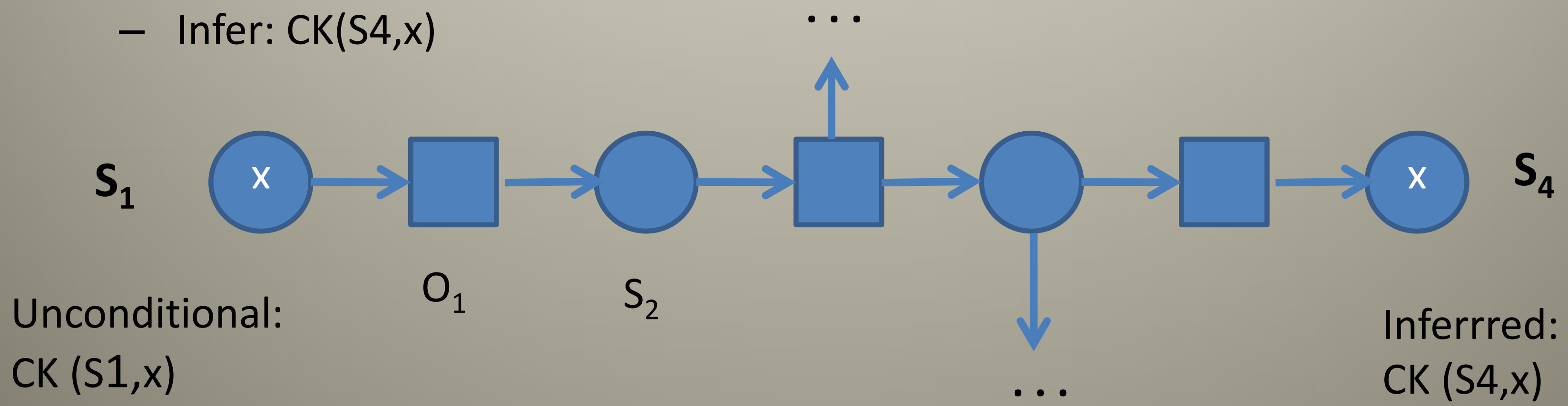
3) *Inference rule* for CS:

$$\exists S\ (CK(S,x) \wedge CW(S,O)) \Rightarrow CS(O,x)$$

Closure property: All CS or CK relationships must be true either unconditionally or by one of the two inference rules.

Derivation Example

- Given: CW(S1,O1), CR(O1,S2), CW(S2,O3) etc. (access control rules)
- Given: CK (S1,x): (unconditional relationship)
- Infer: CS (O1,x)
  - Since CK (S1,x) ∧ CW(S1,O1)
- Infer: CK (S2,x)
  - Since CS (O1,x) ∧ CR (S2,O1)
- Infer: CS (O2,x)
  - Since CK (S2,x) ∧ CW (S2,O2)
- …
- Infer: CK(S4,x)



Formalizing confidentiality and integrity invariants

- Confidentiality invariants express *who can know what*, so they can be expressed in terms of **CK** predicate
- Integrity invariants express *where information can end up* so they can be expressed in terms of **CS** predicate

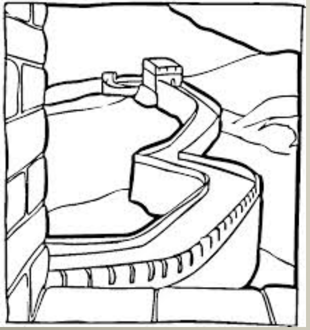
In terms of sets

- CKS(S): (a set) the data that subject S can know
- CSS(O): (a set) the data that object O can store

- Information transfer is irreversible, i.e. once a data item has been included in CKS or CSS it cannot be removed

Labels

- Data variables, Subjects and Objects are labeled to indicate their security status
  - x: TopSecret
  - y: BankAmerica
  - S: {BankAmerica, RoyalBank}



Example: Static Chinese Wall

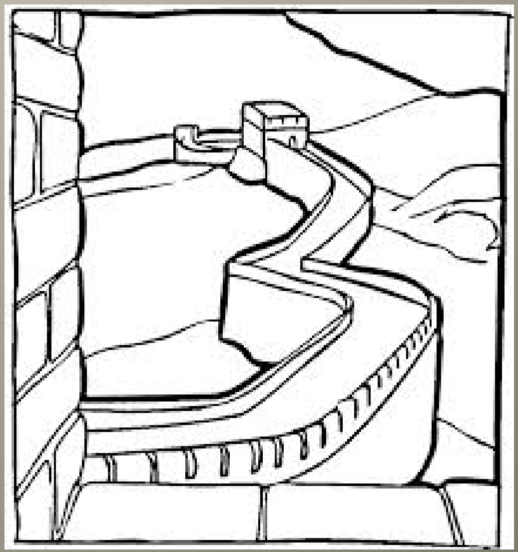
Invariant view

- There are ‘compatible’ and ‘incompatible’ information domains
  - E.g. two banks have incompatible information that must be kept separate
- Invariants:
  - Confidentiality: Subjects are allowed to *know* only compatible information
  - Integrity: Objects are allowed to *store* only compatible information

Example: Static Chinese Wall

Transformation view

- Allowed transformations are:
  - *Subjects* can only *read* from objects with compatible information
  - *Subjects* can only *write* on objects with compatible information



Formalizing Static ChWall

- Security domains:
  - Bank1, Bank2, Oil
  - Compatibility relationship ~
    - Bank1~Oil, Bank2~Oil but **not** Bank1~Bank2
- *Allowed labels* are sets of security domains that contain *only mutually compatible domains*
  - {}, {Bank1}, {Bank2}, {Oil}, {Bank1, Oil}, {Bank2, Oil}

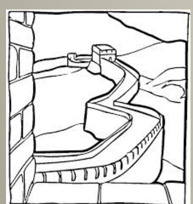
Allowed transformations for ChWall

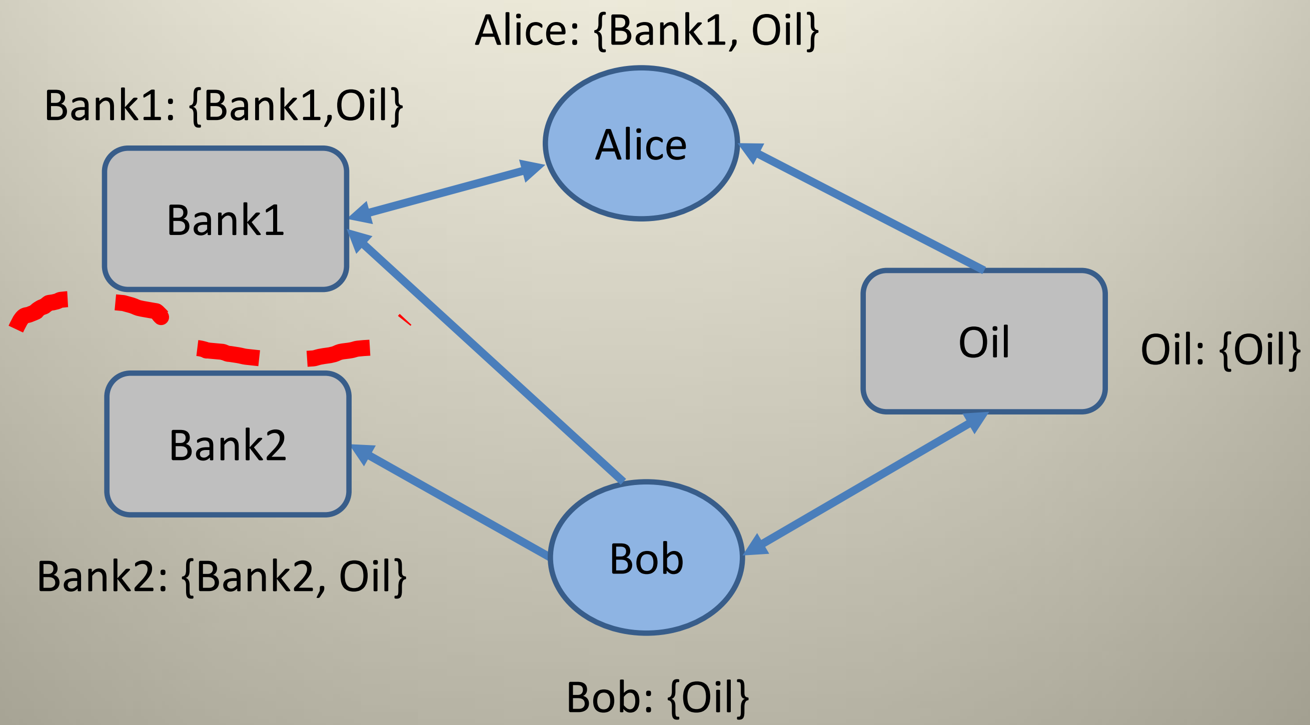
(Access Control rules)

- $CR(S:\Delta, O:\Delta') \leftrightarrow \Delta' \subseteq \Delta$
  - a subject can read from an object iff the object can contain only data variables that the subject can know
- $CW(S:\Delta, O:\Delta') \leftrightarrow \Delta \subseteq \Delta'$
  - a subject can write on an object iff the subject can know only data variables that the object can store

- The result is that incompatible information is not allowed to cross the ChWall

ChWall Example





- This label assignment is one of several that enforce ChWall between Bank1 and Bank2
- Arrows show resulting CR, CW relationships

Formal Invariant Properties for ChWall

- Δ set of allowed labels
- Confidentiality:
  - $x{:}D \in CKS(S{:}\Delta) \leftrightarrow D \in \Delta$
    - E.g. x:Bank1 cannot be known by S:{Bank2,Oil}
      - Invariant could be violated only for subjects containing both Bank1 and Bank2 in their labels: not allowed
- Integrity:
  - $x{:}D \in CSS(O{:}\Delta) \leftrightarrow D \in \Delta$
    - E.g. x:Bank1 cannot be stored in O:{Bank2,Oil}
      - Similar reason

Proving ChWall invariants

- So it is easy to prove that, given the set of allowed transformations, the invariant properties for CWall hold
  - E.g. that x:Bank1 will never end up in O:{…Bank2…}
  - Since labels including {Bank1, Bank2} cannot exist

Proof technique

- Our proofs are based on the following simple induction principle:
  - Suppose that a property P is true for some set
  - And suppose that there are rules for adding elements to the set, which check whether P will still be true after the addition
  - Then obviously P will remain true in the set
    - So P is *invariant* with respect to adding information to a set of acquired information

Dynamic systems

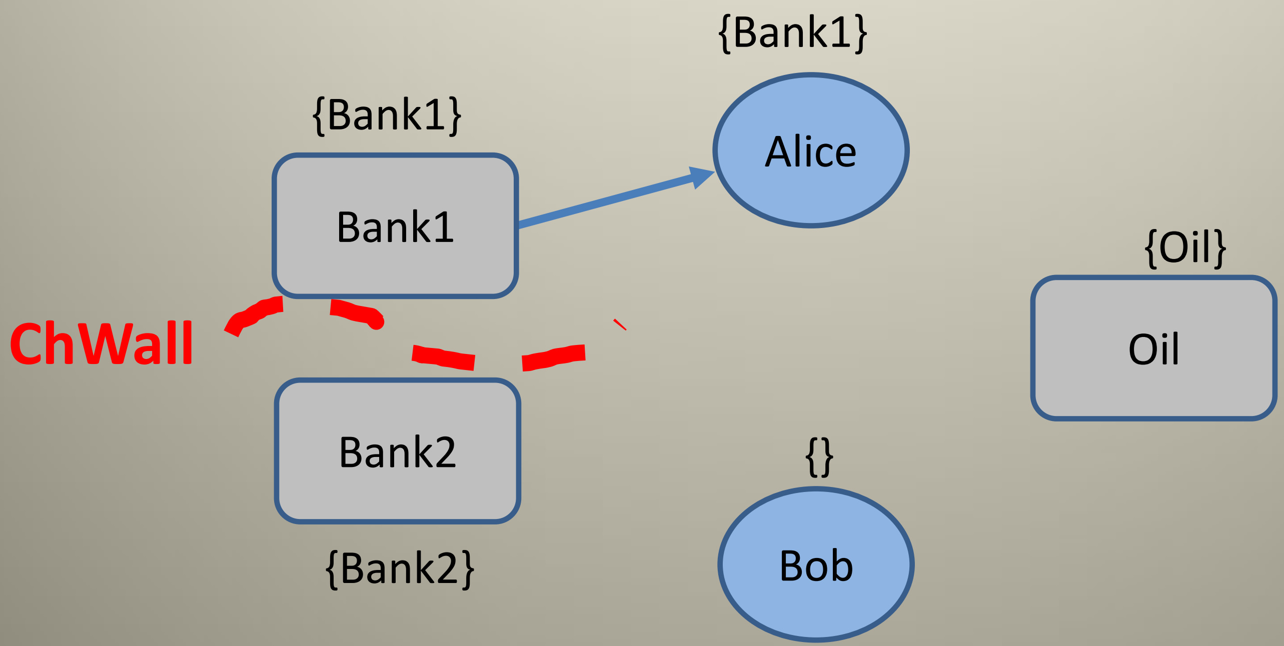
- So far, labels were fixed
  - Our ChWall is a simplification so far
- In dynamic systems, *labels change* as the system progresses
  - E.g. in real ChWall,
    - Labels of subjects change as they read new objects
      - They can now know new information
    - Labels of objects change as more things are stored in them
      - They can now store new information

Dynamic ChWall

- Standard ChWall is dynamic:
  - At the beginning, any subject can read from or write to any object
  - These operations alter the labels and the sets CKS and CSS, thus changing the compatibility relationships between subjects and objects hence the CR or CW relationships
  - But labels with incompatible information are still not allowed

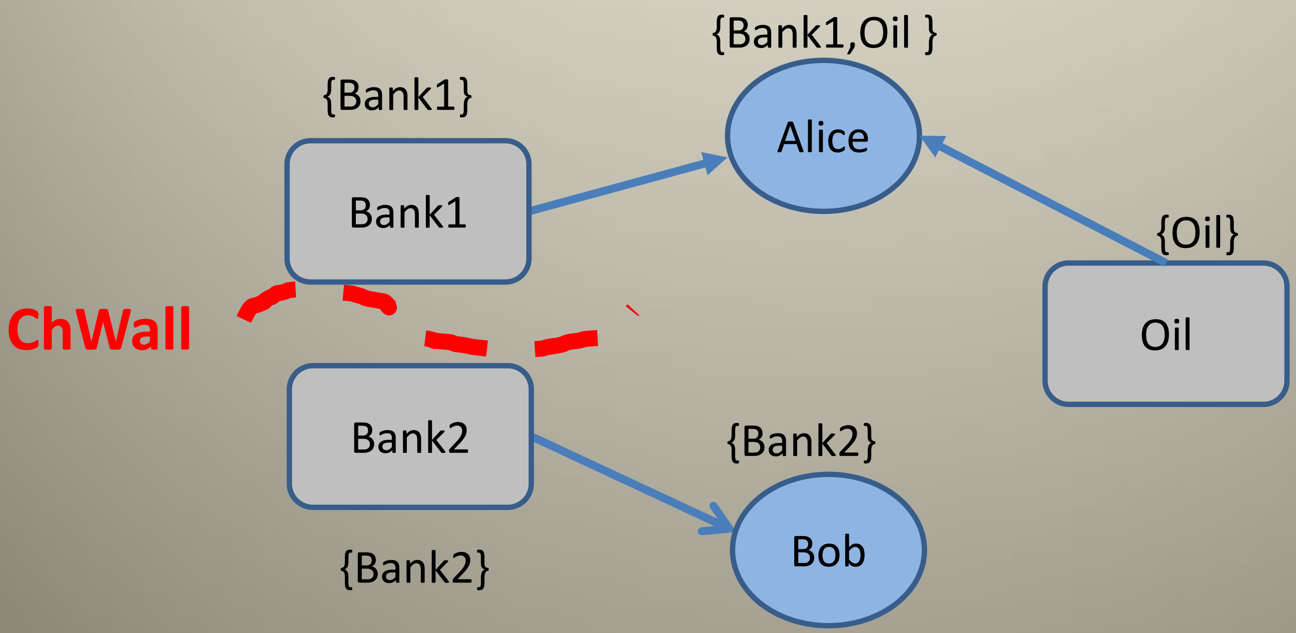
Example

- Initial state:
  - Alice:{}; Bob:{}; Bank1:{Bank1}; Bank2:{Bank2}; Oil:{Oil}
- Alice Reads from Bank1, now Alice: {Bank1}



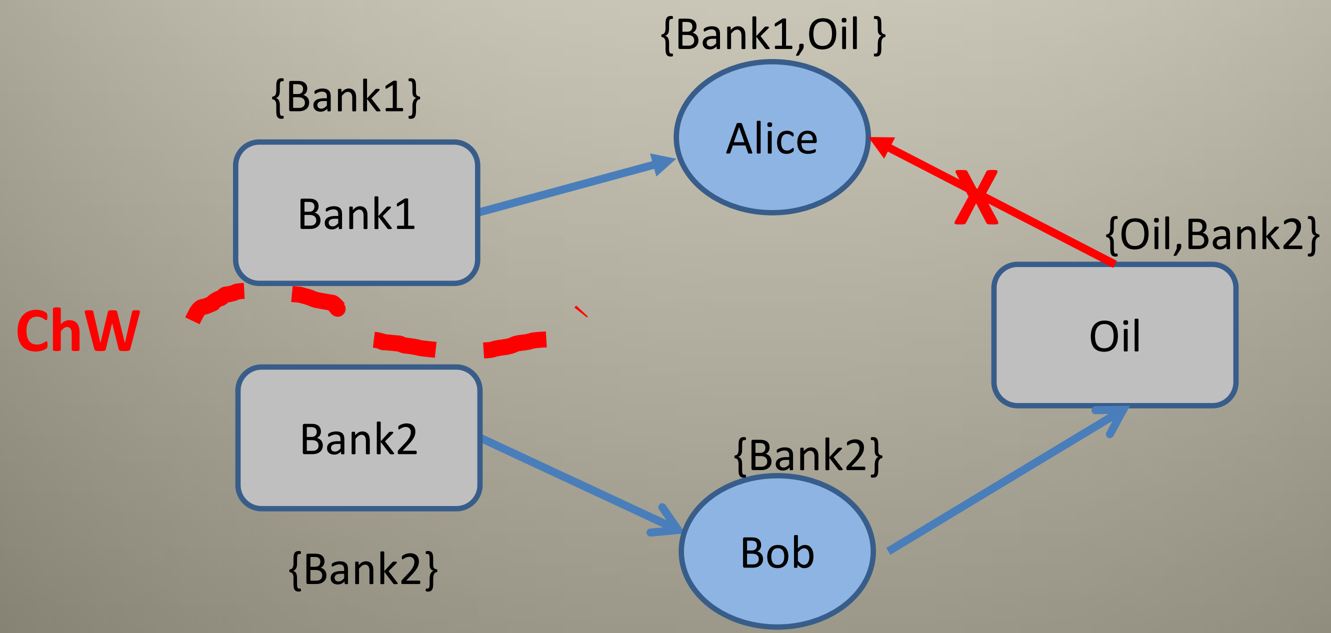
Dynamic ChWall Example

- Initial state:
  - Alice:{}; Bob:{}; Bank1:{Bank1}; Bank2:{Bank2}; Oil:{Oil}
- Alice Reads from Bank1, now Alice: {Bank1}
- Bob Reads from Bank2, now Bob: {Bank2}
- Alice Reads from Oil, now Alice:{Bank1,Oil }



Example

- Initial state:
  - Alice:{}; Bob:{}; Bank1:{Bank1}; Bank2:{Bank2}; Oil:{Oil}
- Alice Reads from Bank1, now Alice: {Bank1}
- Bob Reads from Bank2, now Bob: {Bank2}
- Alice Reads from Oil, now Alice:{Bank1,Oil }
- Bob writes on Oil, now Oil: {Oil,Bank2}
- ¬(Bank1~Bank2) so labels containing both are not allowed
- Future attempts of Alice to read from or write to Oil are blocked



The construction

- We introduce Read and Write operations
- If executed when CR or CW are false they cause state changes
- New states are characterized by new label assignments, reflecting the new CK and CW relationships
- However *Read and Write operations that lead to disallowed labels are not possible*
- So at some point all allowed labels will be used
  - The system becomes stabilized
- Go to ‘static ChWall’ case

Summary of results 1

- We have introduced a new method for reasoning about properties of access control systems
  - Formalizing intuitive concepts
- We have shown its applicability to a number of classical access control models:
  - Bell-La Padula, Biba, Lattice-Based, RBAC, High-Water Mark, Chinese Wall
  - These models were very simplified but there is no real obstacle to extending the reasoning to the full models

Summary of results 2

- This single method has been shown to be appropriate for proving several data flow properties of these models
  - Conventional presentations use different methods for each model
  - Proofs are simple and intuitive

Developments

New access control methods

- Our reasoning method allows to decompose the classical methods into elementary constituents
- This leads to the discovery of new elementary access control methods, that can be combined in many different ways
- They can be studied with our technique

Future work

- Assess and develop the usefulness of the technique with respect to
  - more realistically described access control models of various kinds
  - automatic theorem proving
  - model combinations
    - → a new life for MAC models?