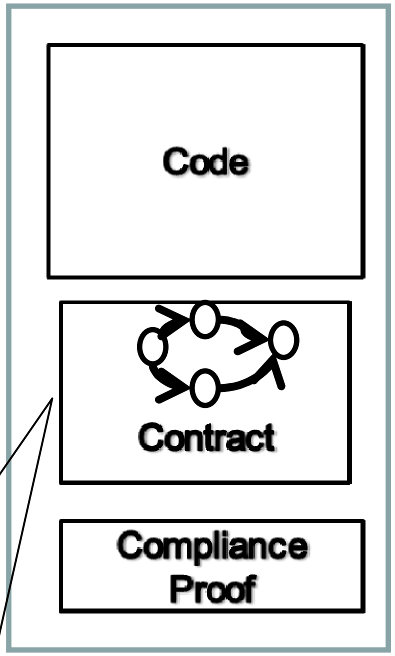
Università degli Studi di Trento

Security-by-Contract using Automata Modulo Theory (AMT)

Ida S.R. Siahaan

Is an application trustworthy ?

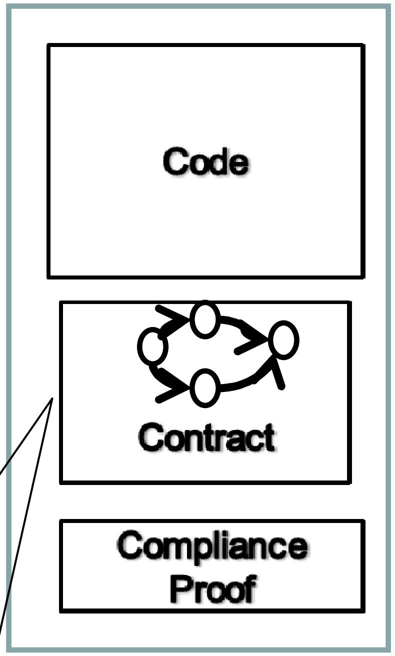


Contract:

specification of application's behavior concerning security-relevant actions

Is an application trustworthy ?

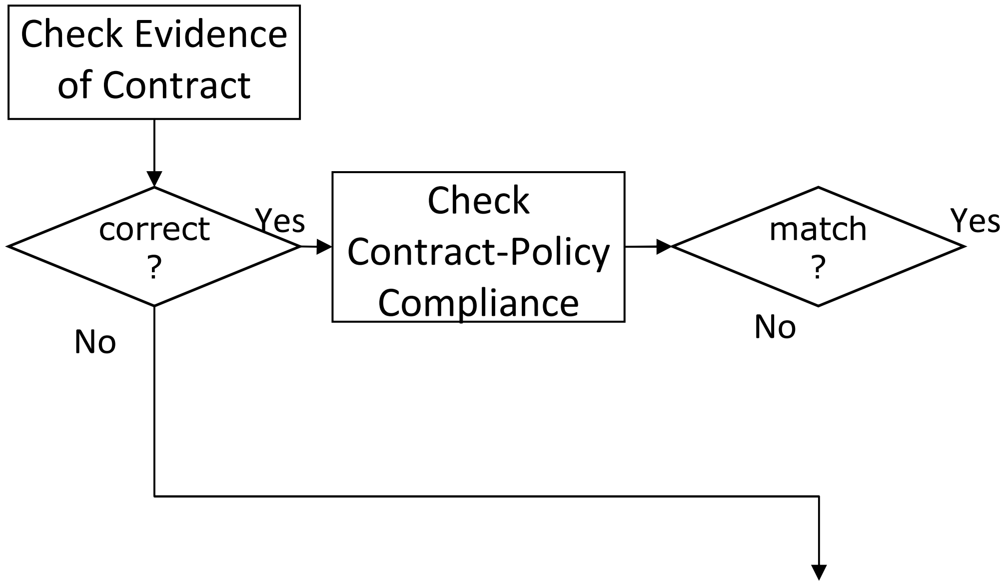
- **Reveal what it does**
  - Design software with security claims
- **Demonstrate its evidence**
  - Check that the application fulfills its claims
- **Verify its compliance**
  - Compliance of Contracts with Policies
- **Assurance for trustworthiness**
  - Inline security policy into the application
  - Run-time monitor the services

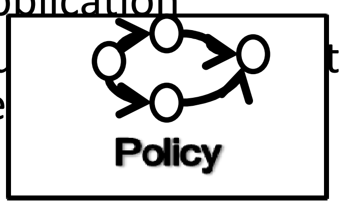


Contract:
specification of application's behavior concerning security-relevant actions

Is an application trustworthy ?

- **Reveal what it does**
  - Design software with security claims
- **Demonstrate its evidence**
  - Check that the application fulfills its claims
- **Verify its compliance**
  - Compliance of Contracts with Policies
- **Assurance for trustworthiness**
  - Inline security policy into the application
  - Ru… the se…



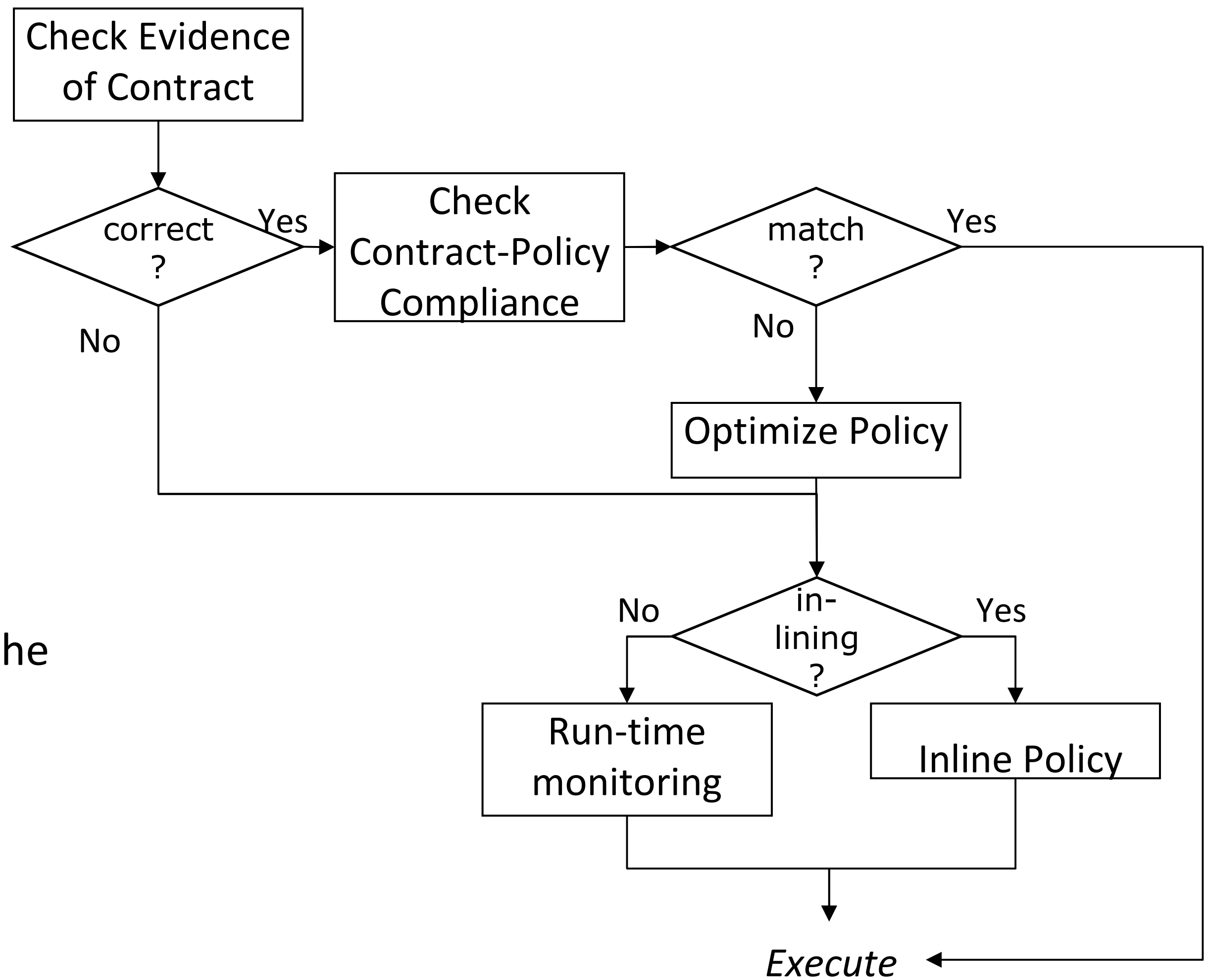


Policy:
specification of application's acceptable behavior to be executed on a platform concerning security-relevant actions

Università degli Studi di Trento

Is an application trustworthy ?

- **Reveal what it does**
  - Design software with security claims
- **Demonstrate its evidence**
  - Check that the application fulfills its claims
- **Verify its compliance**
  - Compliance of Contracts with Policies
- **Assurance for trustworthiness**
  - Inline security policy into the application
  - Run-time monitor the services



Road Map

Security-by-Contract

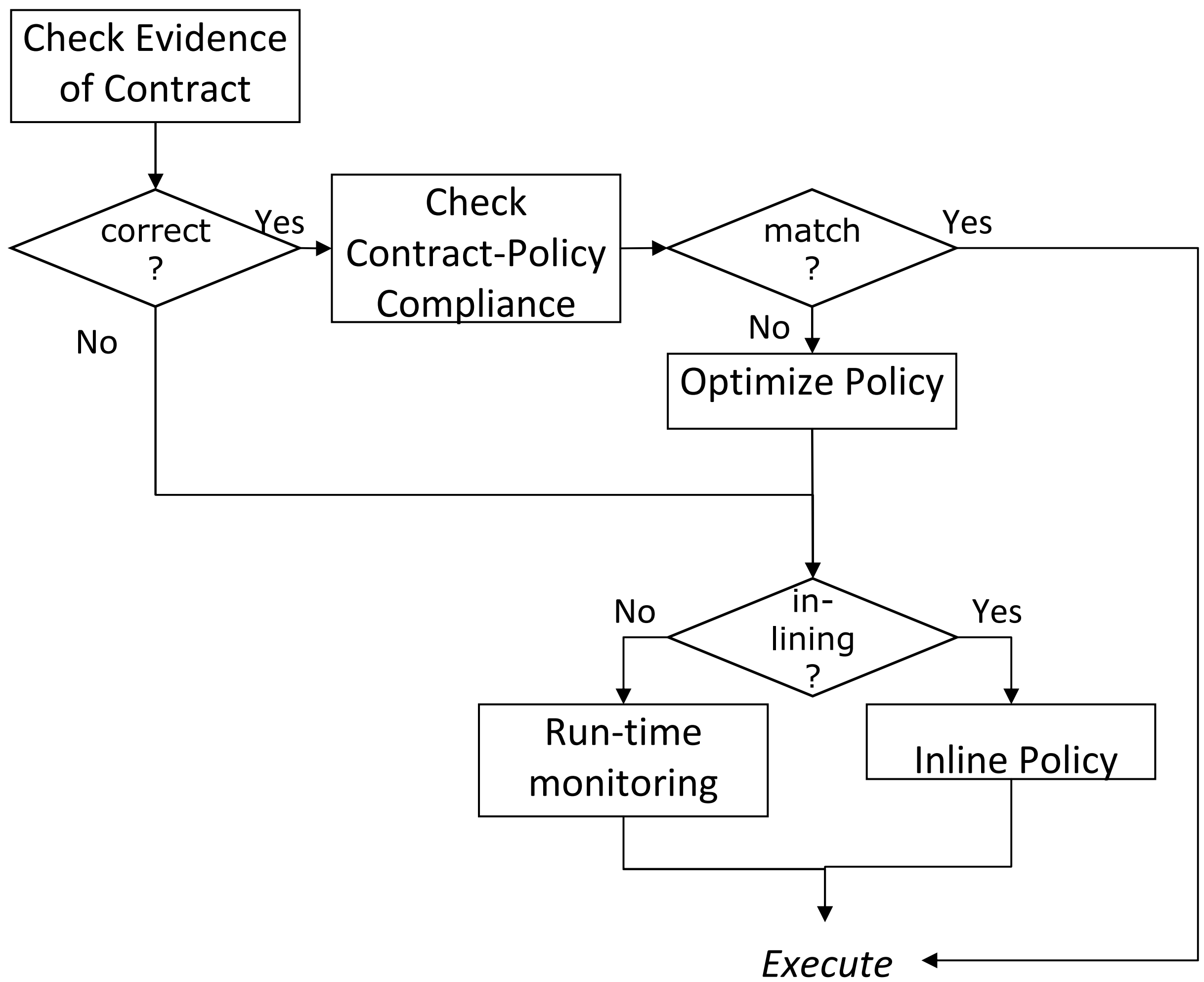
Automata Modulo Theory

On-the-fly Matching

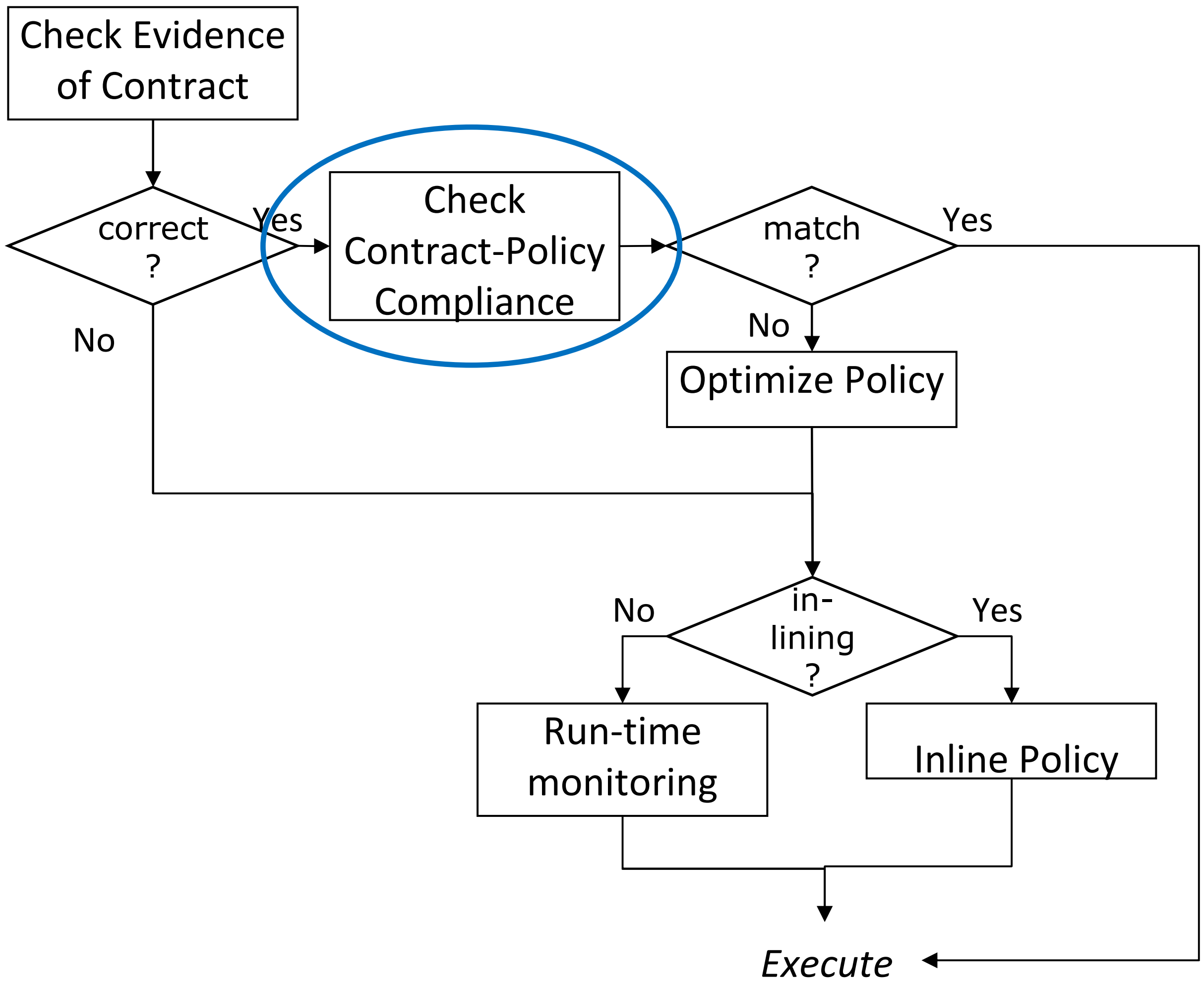
Simulation Matching

IRM Optimization

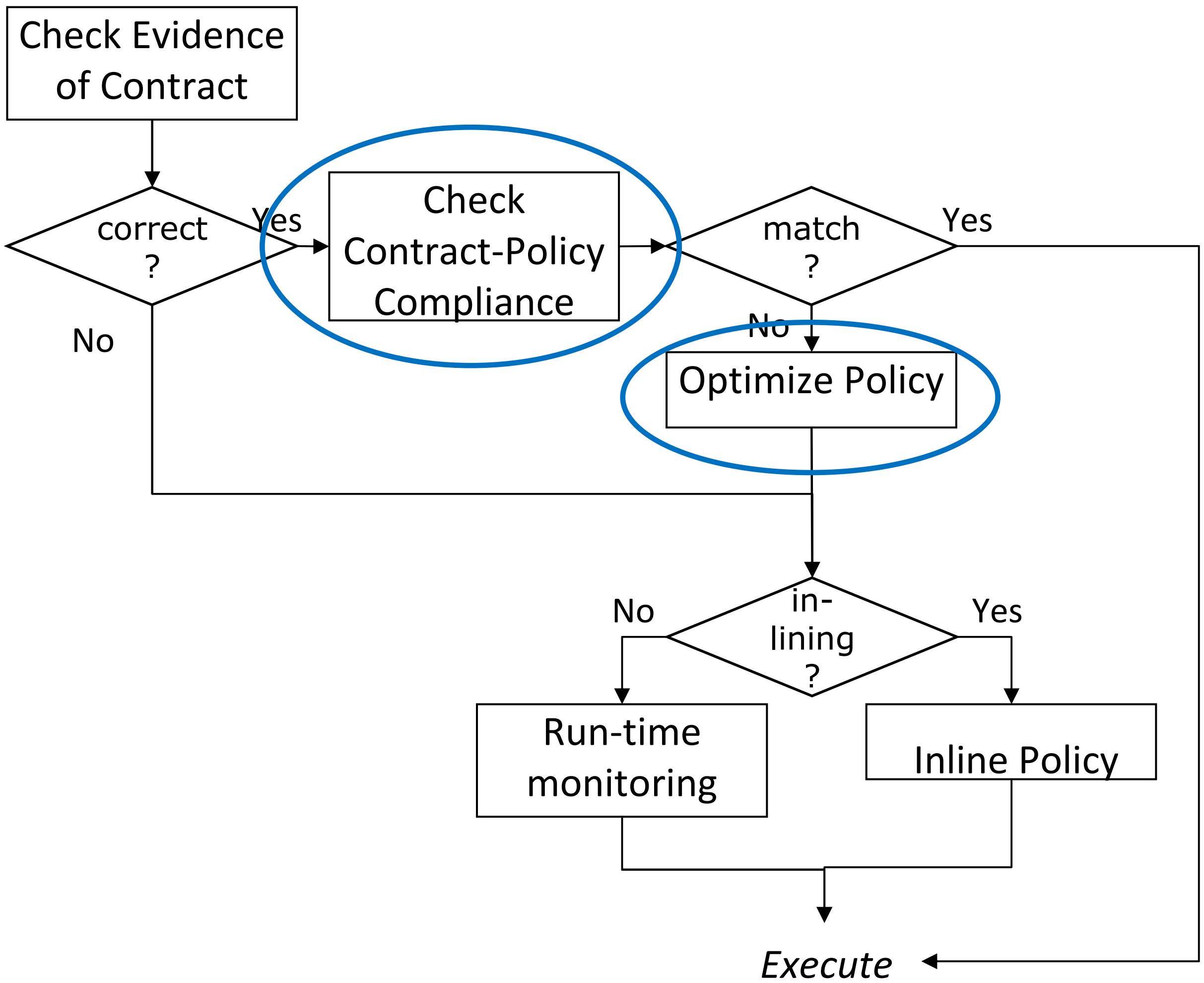
Thesis Works



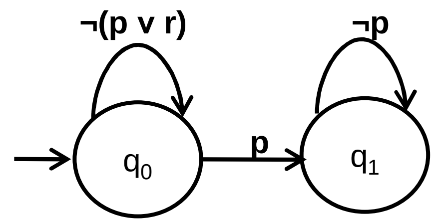
Thesis Works



Thesis Works



Why not Security Automata ?



- **Class of Büchi automata accepting safety properties (recognizers) [Schneider-TISSec'00]**
  - a countable set Q of *automaton states,*
  - a countable set I of *input symbols*
  - a *transition function* $\delta : (Q \times Q) \rightarrow 2^Q$, and
  - a countable set $Q_0 \subseteq Q$ of *initial automaton states*

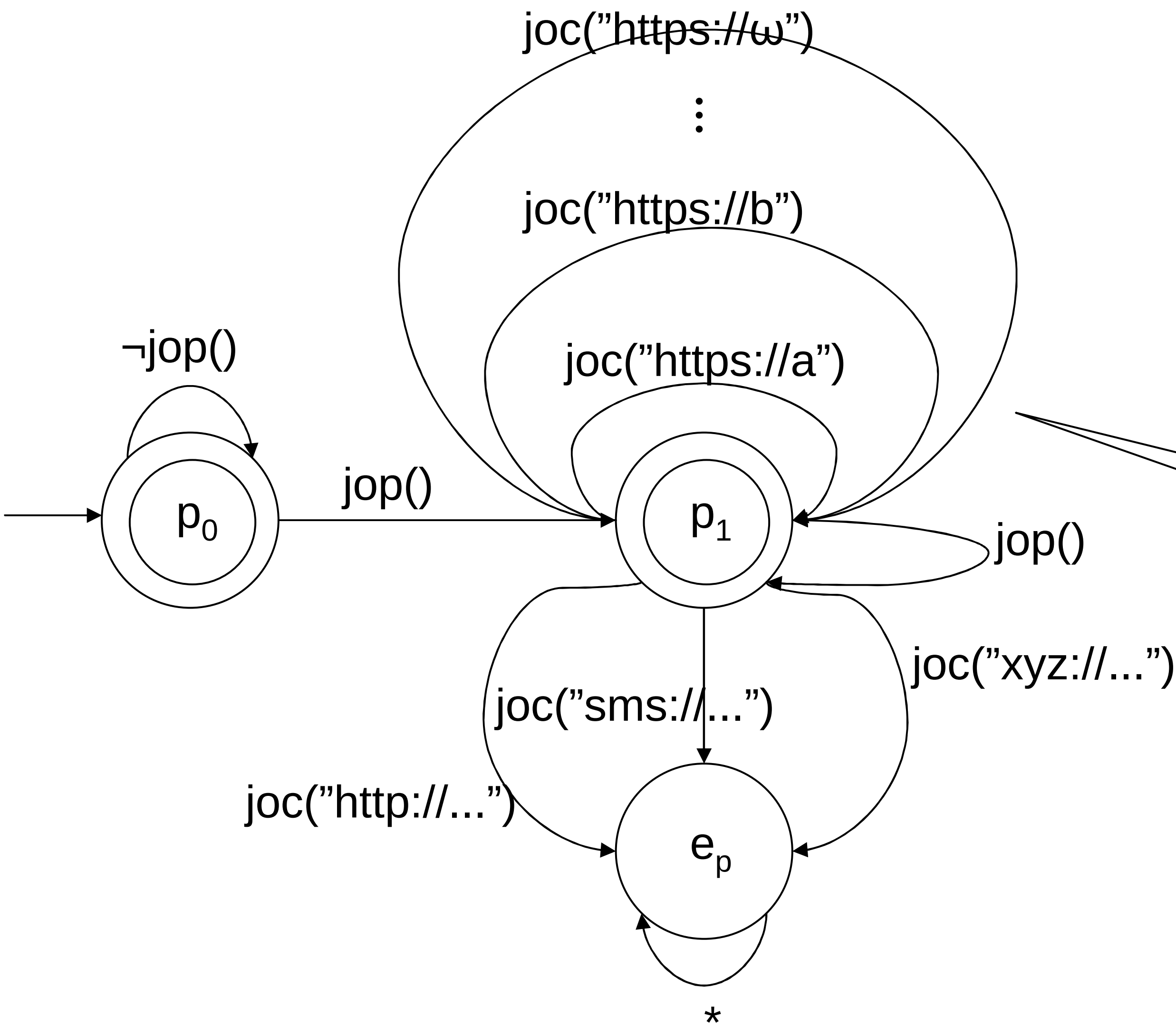
Liveness Property

- **It is rare, but it exists**
  - Example: A security requirement for banking applets
    - an application should use all the permissions it requires
    - to avoid over-entitlement which can be the source of potential (and possibly unknown) attacks

Infinite Transitions

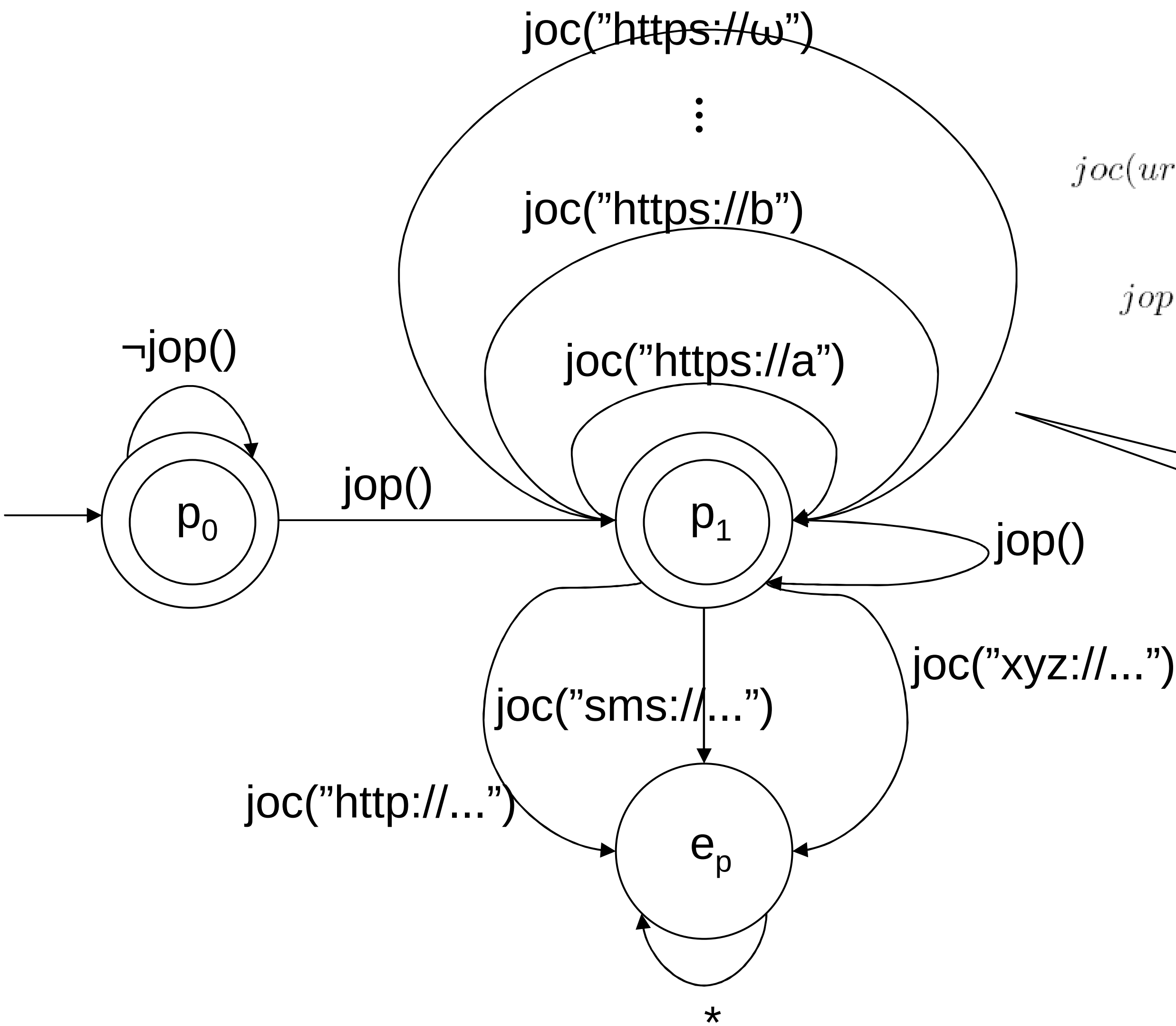
Example of a Policy:
"After PIM is accessed only secure connections can be opened''

Infinite Transitions



Example of a Policy:
"After PIM is accessed only secure connections can be opened''

Infinite Transitions

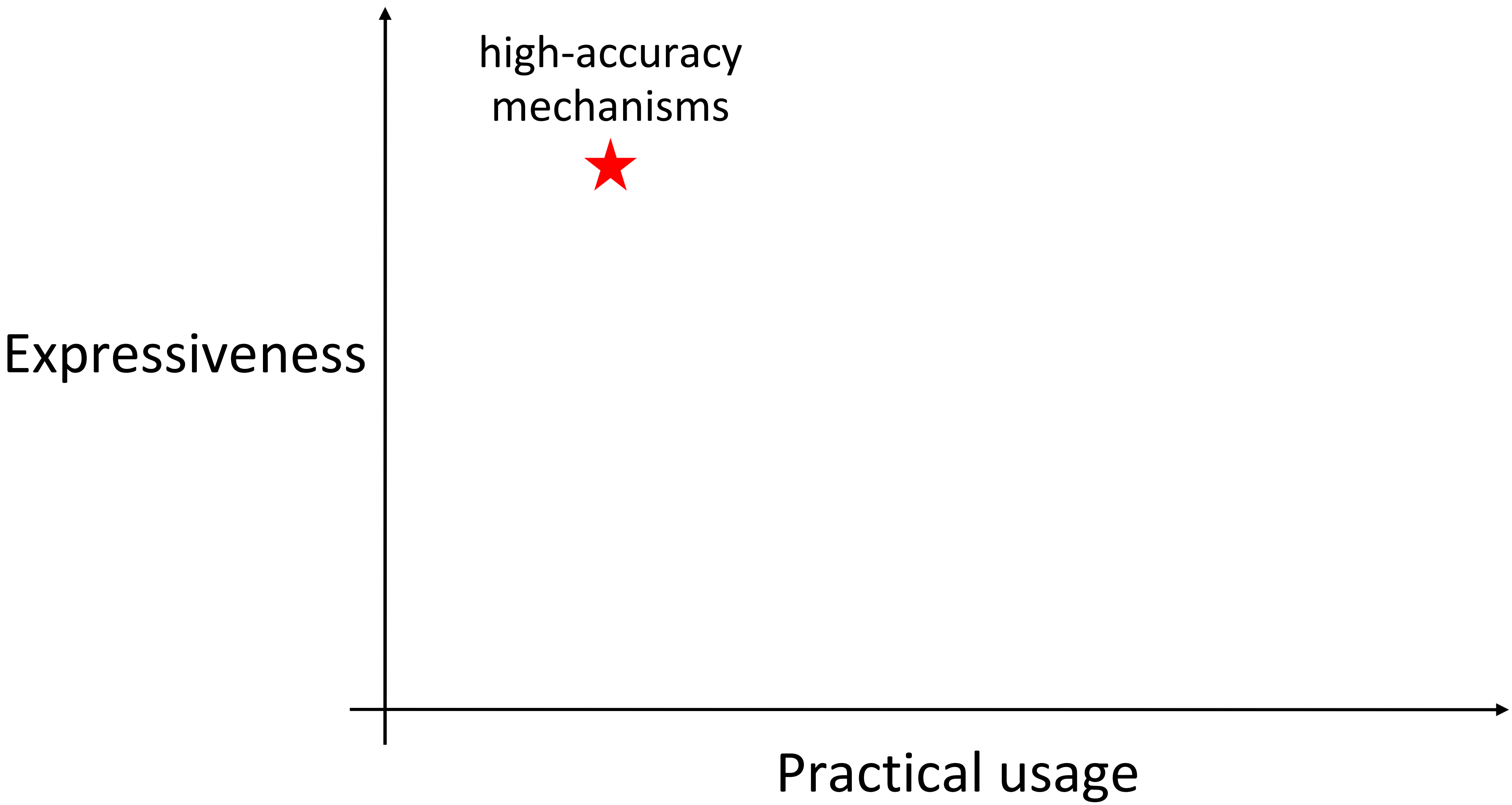


$$joc(url) \doteq \texttt{javax.microedition.io.Connector.open}(url)$$

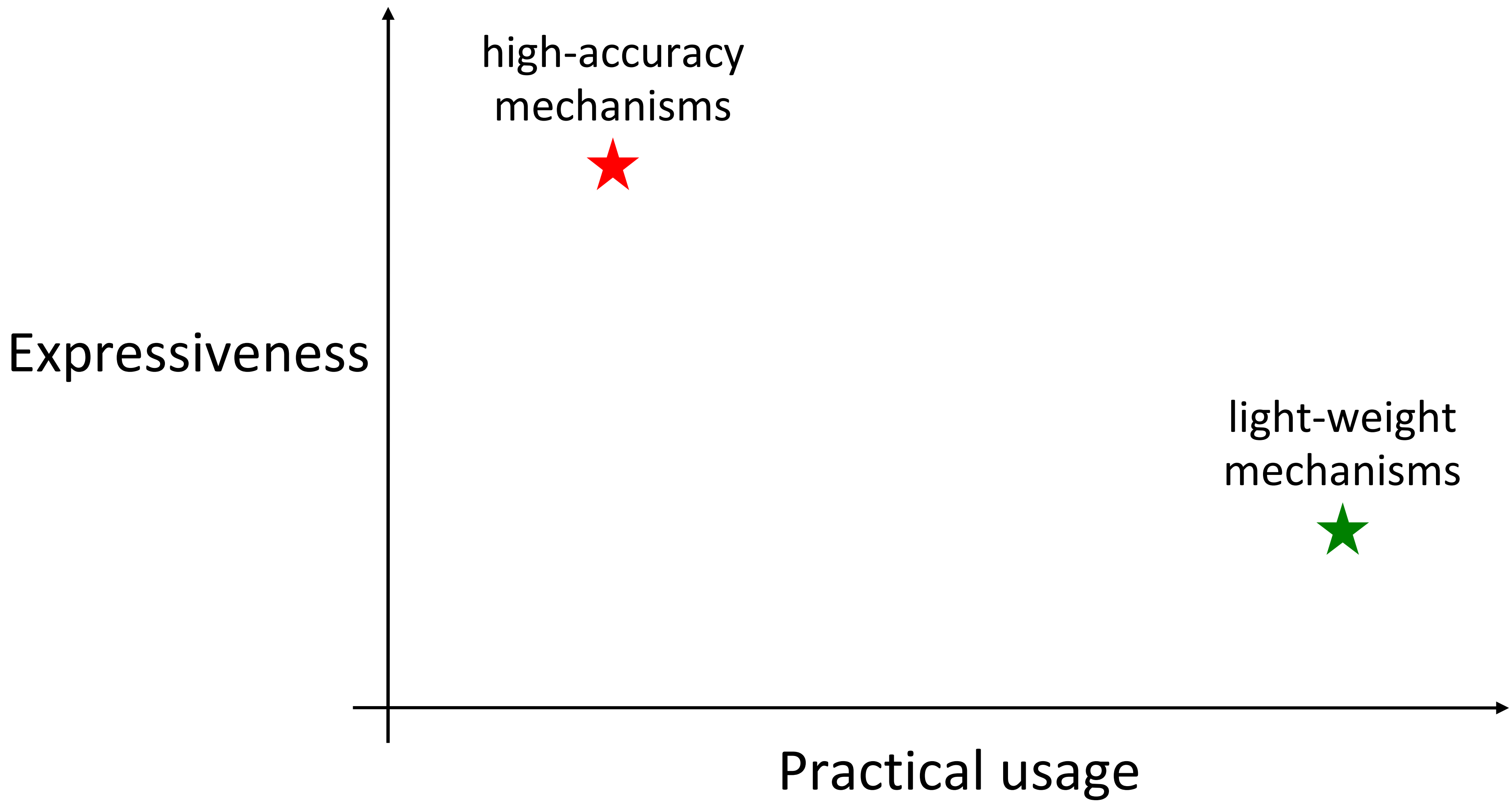
$$jop() \doteq \texttt{javax.microedition.pim.PIM.openPIMList}(\ldots)$$

Example of a Policy:
"After PIM is accessed only secure connections can be opened"

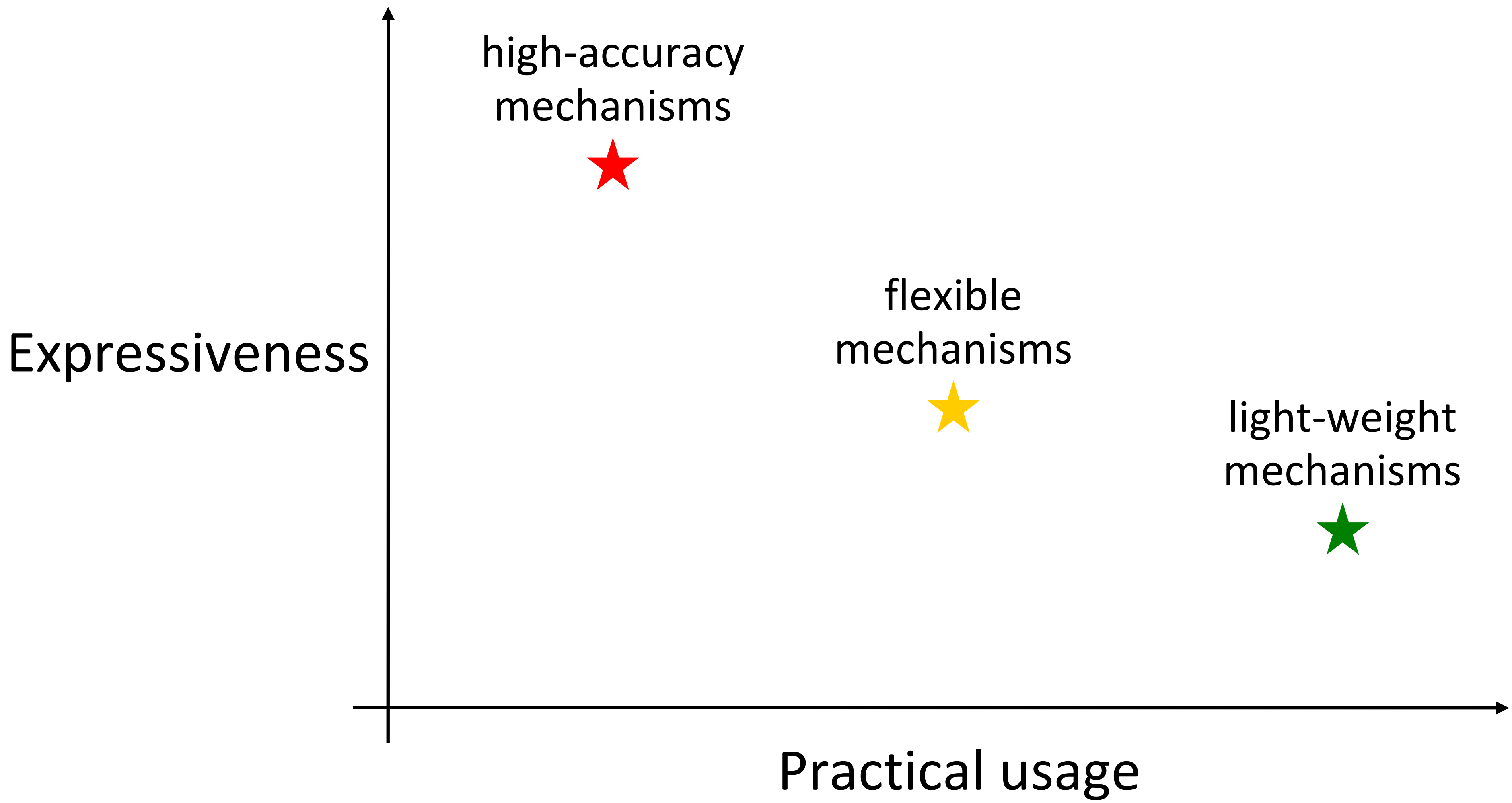
Security Policies Enforcement Mechanisms



Security Policies Enforcement Mechanisms



Security Policies Enforcement Mechanisms



Road Map

Security-by-Contract

Automata Modulo Theory

On-the-fly Matching

Simulation Matching

IRM Optimization

Automata Modulo Theory (AMT)
as flexible mechanism

AMT = Büchi automata + Satisability Modulo Theories

Satisability Modulo Theories (SMT) [Sebastiani-JSAT'07]

- The problem of deciding the satisfiability of a first-order formula with respect to some decidable first-order theory T (SMT(T))
  - **A Σ-theory is a set of first-order sentences with signature Σ**
- Examples of theories of interest:
  - **Equality and Uninterpreted Functions (EUF),**
  - **Linear Arithmetic (LA): both over the reals (LA(Q)) and the integers (LA(Z))**
  - **Combination of two or more theories $T_1,\ldots,T_n$.**
- Examples of SMT tools:
  - **Z3, MathSAT**

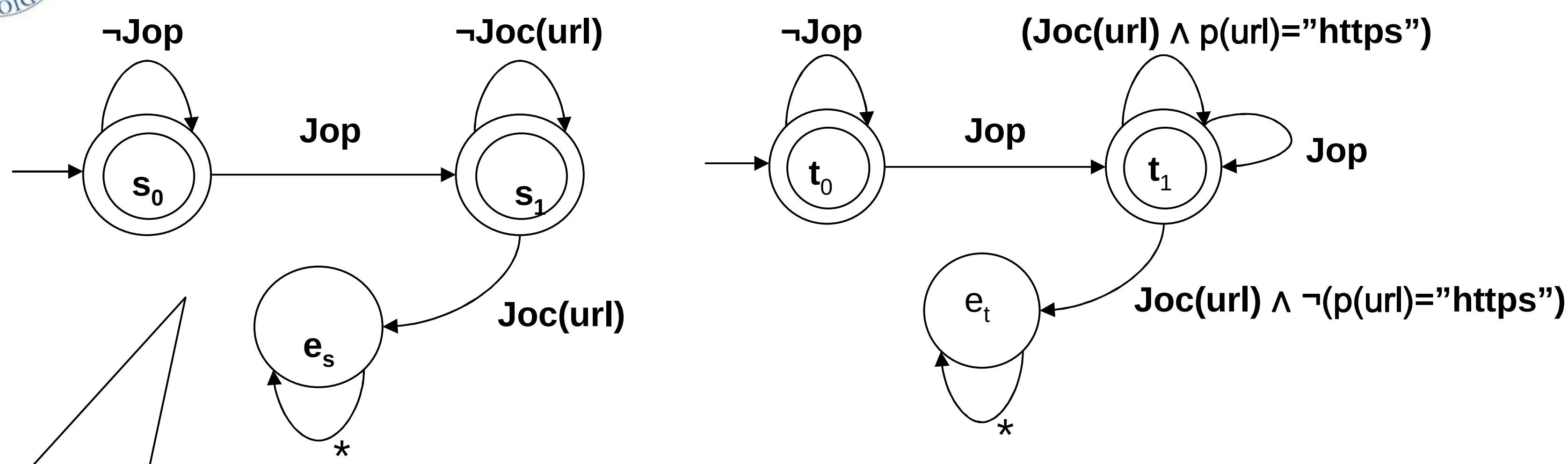
Automata Modulo Theory (AMT)

- **Let A = < S, Σ, $\mathcal{T}$, $\mathcal{E}$, Δ, s_0, F> be an AMT [MS-NordSec'07]**
  - a finite set S of *automaton states*,
  - a set $\mathcal{E}$ of formulae in the language of the Σ-Theory $\mathcal{T}$ as *input symbols,*
  - an *initial state* $s_0 \in S$,
  - a set $F \subseteq S$ of *accepting states*, and
  - a *labeled transition relation* $\Delta \subseteq S \times \mathcal{E} \times S$

Examples of AMT

Example of a Contract
"After PIM is opened no connections are allowed"

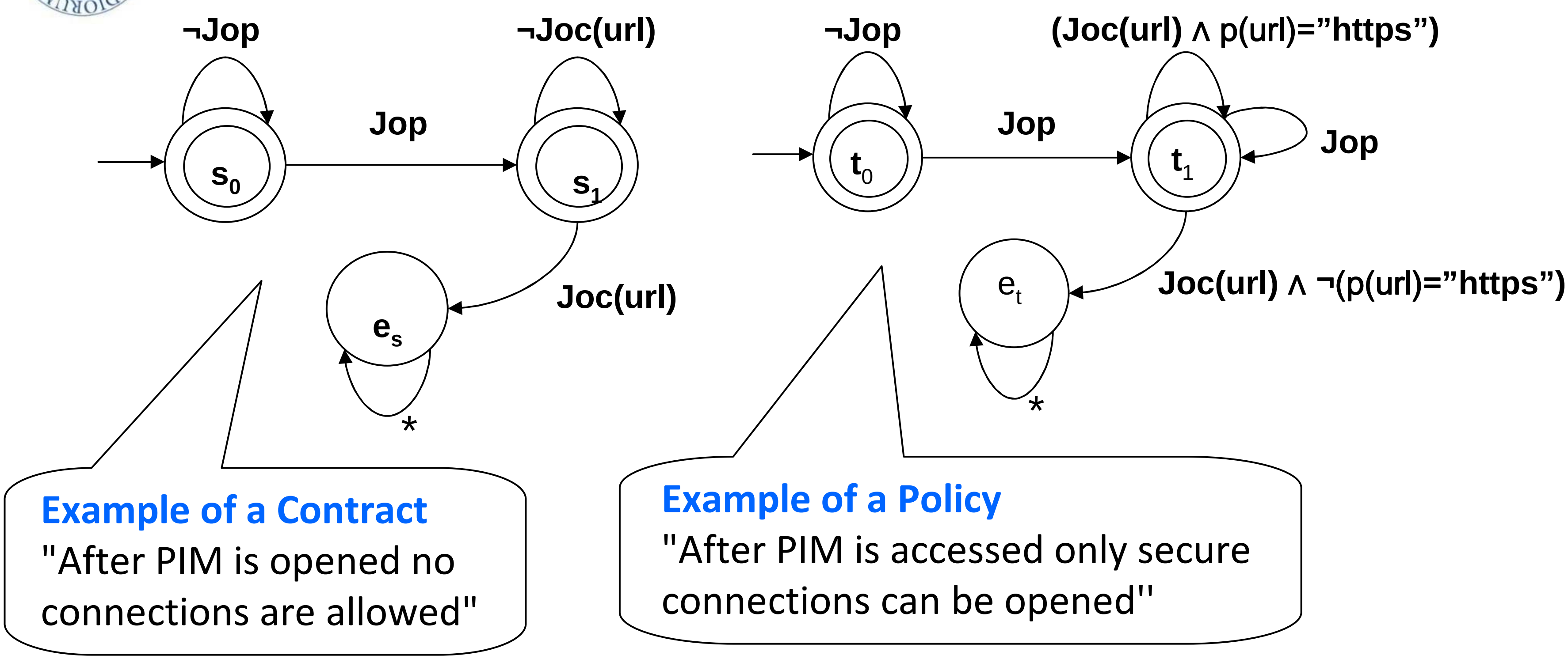
Examples of AMT



Example of a Contract
"After PIM is opened no connections are allowed"

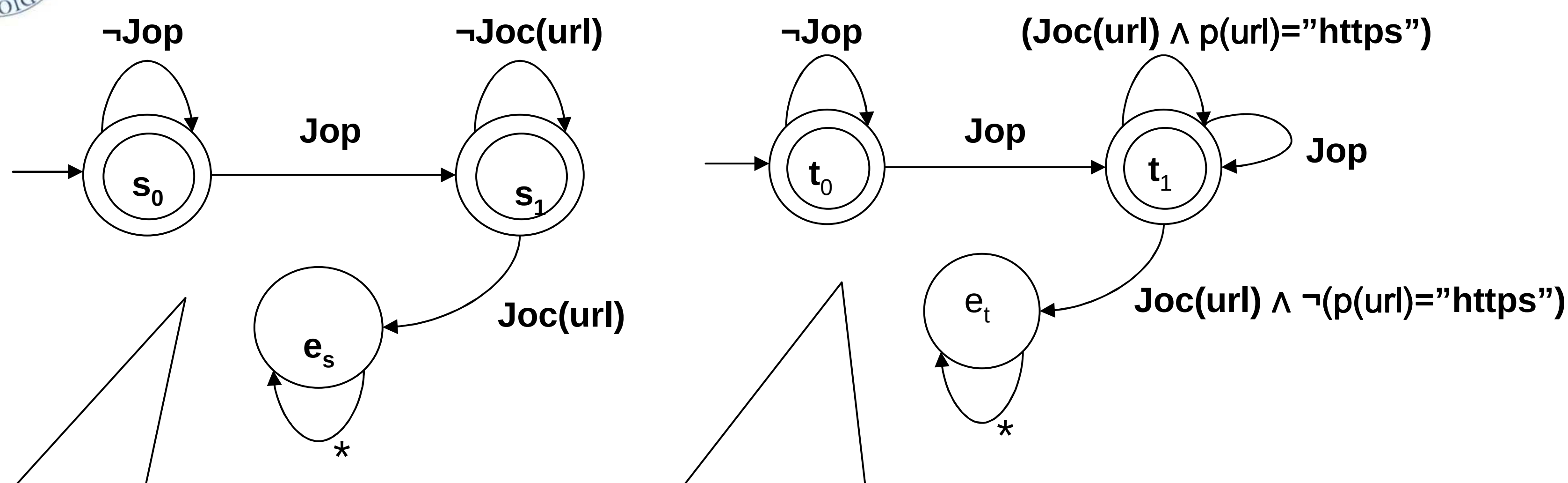
$$
\begin{aligned}
Joc(url) &\doteq \texttt{Joc(joc,url)} \\
Jop &\doteq \texttt{Jop(jop,}x_1,\ldots,x_n\texttt{)} \\
p(url) = type &\doteq url.\texttt{startsWith}(type) \\
joc &\doteq \texttt{javax.microedition.io.Connector.open} \\
jop &\doteq \texttt{javax.microedition.pim.PIM.openPIMList}
\end{aligned}
$$

Examples of AMT



$$
\begin{aligned}
Joc(url) &\doteq \texttt{Joc(joc,url)} \\
Jop &\doteq \texttt{Jop(jop,}x_1,\ldots,x_n\texttt{)} \\
p(url) = type &\doteq url\texttt{.startsWith}(type) \\
joc &\doteq \texttt{javax.microedition.io.Connector.open} \\
jop &\doteq \texttt{javax.microedition.pim.PIM.openPIMList}
\end{aligned}
$$

Examples of AMT



Example of a Contract

"After PIM is opened no connections are allowed"

Example of a Policy

"After PIM is accessed only secure connections can be opened"

$$
\begin{aligned}
Joc(url) &\doteq \texttt{Joc(joc,url)} \\
Jop &\doteq \texttt{Jop(jop,}x_1,\ldots,x_n\texttt{)} \\
p(url) = type &\doteq url\texttt{.startsWith}(type) \\
joc &\doteq \texttt{javax.microedition.io.Connector.open} \\
jop &\doteq \texttt{javax.microedition.pim.PIM.openPIMList}
\end{aligned}
$$

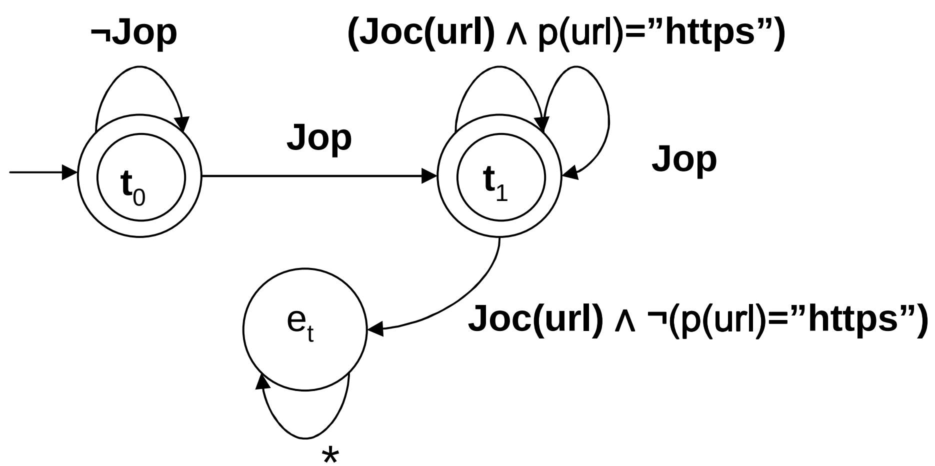
Symbolic Run in AMT

- **Let A = < S, Σ, $\mathcal{T}$, $\mathcal{E}$, Δ, s_0, F> be an AMT**
- **A *symbolic run* of A is a sequence of states alternating with expressions σ = < $q_0e_1q_1e_2q_2$ ... >:**
  - $q_0 = s_0$
  - $(q_i, e_{i+1}, q_{i+1}) \in \Delta$ and e_{i+1} is $\mathcal{T}$-satisfiable:
    - that is there exists some valuation v over Σ and $\mathcal{T}$ s.t. $v \models e_{i+1}$
    - valuation v is a pair $(\mathfrak{M}, \alpha)$: $\mathfrak{M}$ a model of $\mathcal{T}$ and α an assignment
  - *Finite symbolic run* σ = < $q_0e_1q_1e_2q_2 \ldots q_n$>
  - *Infinite symbolic run* σ = < $q_0e_1q_1e_2q_2 \ldots$ >
- ***Accepting symbolic run:***
  - Finite run: $q_n \in F$
  - Infinite run: there exists some k s.t. $q_k \in F$ and q_k is visited infinitely often

Concrete Run in AMT

- **Let $A = < S, \Sigma, \mathcal{T}, \mathcal{E}, \Delta, s_0, F>$ be an AMT**
- **A *concrete run* of *A* is a sequence of states alternating with valuations σ = < $q_0 v_1 q_1 v_2 q_2$ ... >:**
  - $q_0 = s_0$
  - there exists $e_{i+1} \in \mathcal{E}$:
    - $(q_i, e_{i+1}, q_{i+1}) \in \Delta$
    - there exists some valuation v over Σ and $\mathcal{T}$ *s.t.* $v \models e_{i+1}$
  - *Finite concrete run* σ = < $q_0 v_1 q_1 v_2 q_2$ ... q_n>
  - *Infinite concrete run* σ = < $q_0 v_1 q_1 v_2 q_2$ ... >
- ***Acceptance condition as symbolic run***

Example of an Accepting Run in AMT



Symbolic Run

t0 Jop(jop,file,permission) t1 Joc(joc,url)^p(url)=“https”

t1 Jop(jop,file,permission) t1 Joc(joc,url)^p(url)=“https” …

Concrete Run

t0 (jop,PIM.CONTACT_LIST,PIM.READ_WRITE)

t1 (joc,“https://www.esse3.unitn.it/”)

t1 (jop,PIM.CONTACT_LIST,PIM.READ_ONLY)

t1 (joc,“https://online.unicreditbanca.it/login.htm”) …

Deterministic AMT

- **A = < S, Σ, $\mathcal{T}$, $\mathcal{E}$, Δ, s_0, F> is a deterministic AMT**
  - S, Σ, $\mathcal{T}$, $\mathcal{E}$, s_0, F as before
  - a *labeled transition function* $\Delta \subseteq S \times \mathcal{E} \times S$:
    - for every $s, s_1, s_2 \in S$ and every $e_1, e_2 \in \mathcal{E}$
    - if $(s, e_1, s_1) \in \Delta$ and $(s, e_2, s_2) \in \Delta$ where $s_1 \neq s_2$
    - then $(e_1 \wedge e_2)$ is unsatisfiable in the Σ-Theory $\mathcal{T}$
- **Why determinism matters ?**
  - nondeterministic complementation is complex and exponential blow-up
- **Why considering only the complementation of deterministic automata ?**
  - security policies are naturally deterministic
    - a platform owner should have a clear idea on what to allow or disallow

AMT Complementation and Intersection

- **Complementation:**
  - For each deterministic AMT automaton A there exists a (possibly nondeterministic) AMT that accepts all the words which are not accepted by automaton A.
- **Intersection: Let < S^a, Σ^a, $\mathcal{T}^a$, $\mathcal{E}^a$, Δ^a, s_0^a, F^a> and < S^b, Σ^b, $\mathcal{T}^b$, $\mathcal{E}^b$, Δ^b, s_0^b, F^b> be AMT, the *intersection* automaton *A* = < S, Σ, $\mathcal{T}$, $\mathcal{E}$, Δ, s_0, F> :**
  - $\Sigma = \Sigma^a \cup \Sigma^b$, $\mathcal{T} = \mathcal{T}^a \cup \mathcal{T}^b$, $\mathcal{E} = \mathcal{E}^a \cup \mathcal{E}^b$,
  - $S = S^a \times S^b\ S \times \{1,2\}$, $s_0 = (s_0^a, s_0^b, 1)$, $F = F^a \times S^b \times \{1\}$,
  - for every s ∈ S and for every e ∈ $\mathcal{E}$:

$\Delta = \{<(s^a, s^b, x), (e^a \wedge e^b), (t^a, t^b, y)> \mid (s^a, e^a, t^a) \in \Delta^a$ and $(s^b, e^b, t^b) \in \Delta^b$ and $DecisionProcedure(e^a \wedge e^b) = SAT\}$

$$y = \begin{cases} 2 \text{ if } x = 1 \text{ and } s^a \in F^a \text{ or if } x = 2 \text{ and } s^b \notin F^b \\ 1 \text{ if } x = 1 \text{ and } s^a \notin F^a \text{ or if } x = 2 \text{ and } s^b \in F^b \end{cases}$$

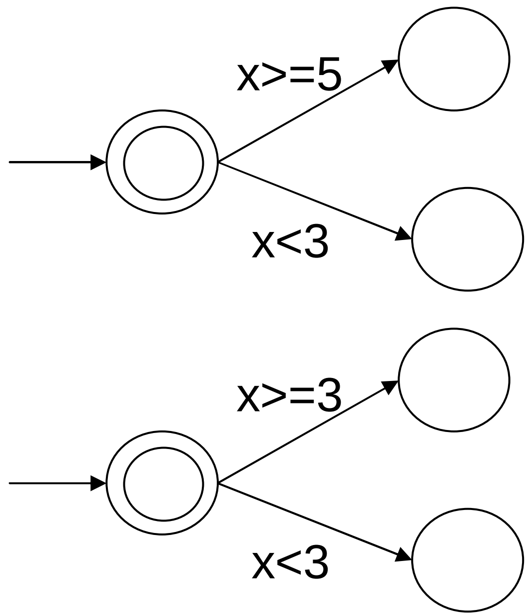
AMT Complementation and Intersection

- **Complementation:**
  - For each deterministic AMT automaton A there exists a (possibly nondeterministic) AMT that accepts all the words which are not accepted by automaton A.
- **Intersection: Let <** $S^a, \Sigma^a, \mathcal{T}^a, \mathcal{E}^a, \Delta^a, s_0^a, F^a$**> and <** $S^b, \Sigma^b, \mathcal{T}^b, \mathcal{E}^b, \Delta^b, s_0^b, F^b$**> be AMT, the *intersection* automaton** A **= <** $S, \Sigma, \mathcal{T}, \mathcal{E}, \Delta, s_0, F$**> :**
  - $\Sigma = \Sigma^a \cup \Sigma^b$, $\mathcal{T} = \mathcal{T}^a \cup \mathcal{T}^b$, $\mathcal{E} = \mathcal{E}^a \cup \mathcal{E}^b$,
  - $S = S^a \times S^b\ S \times \{1,2\}$, $s_0 = (s_0^a, s_0^b, 1)$, $F = F^a \times S^b \times \{1\}$,
  - for every $s \in S$ and for every $e \in \mathcal{E}$:

$$\Delta = \{<(s^a, s^b, x), (e^a \wedge e^b), (t^a, t^b, y)> \mid (s^a, e^a, t^a) \in \Delta^a \text{ and } (s^b, e^b, t^b) \in \Delta^b \text{ and } \boxed{DecisionProcedure(e^a \wedge e^b) = SAT}\}$$

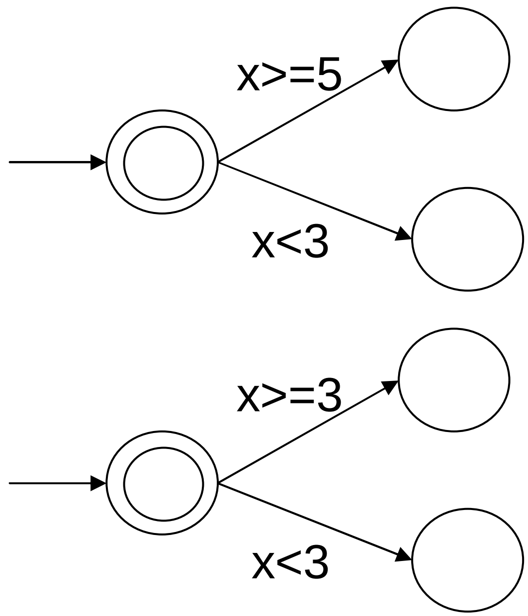
$$y = \begin{cases} 2 \text{ if } x = 1 \text{ and } s^a \in F^a \text{ or if } x = 2 \text{ and } s^b \notin F^b \\ 1 \text{ if } x = 1 \text{ and } s^a \notin F^a \text{ or if } x = 2 \text{ and } s^b \in F^b \end{cases}$$

AMT Intersection

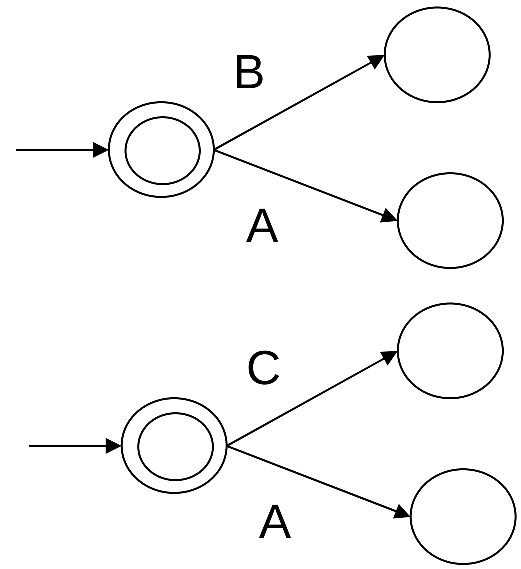


(a) Example of Automata

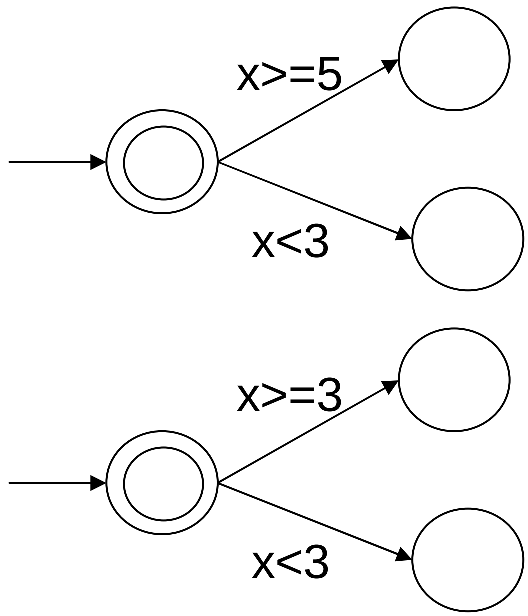
AMT Intersection



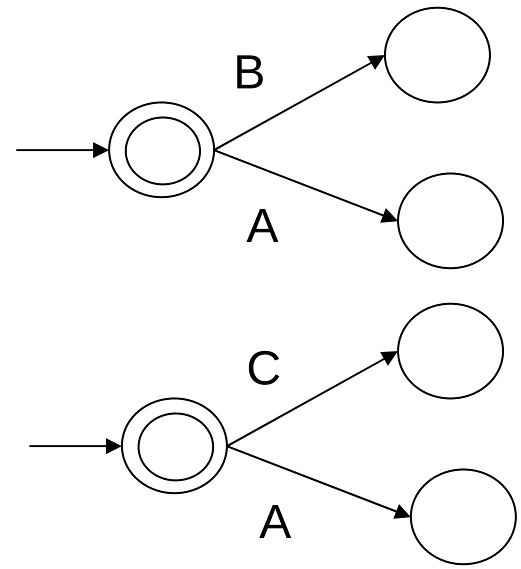
(a) Example of Automata



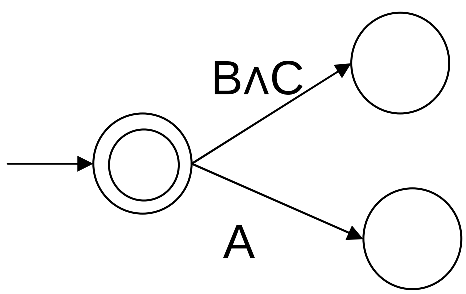
AMT Intersection



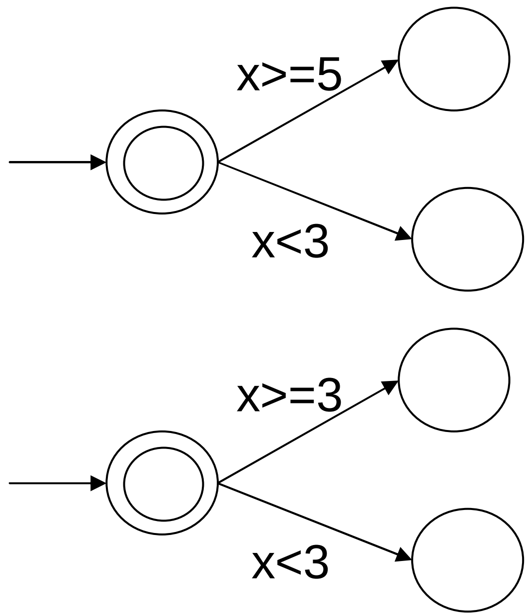
(a) Example of Automata



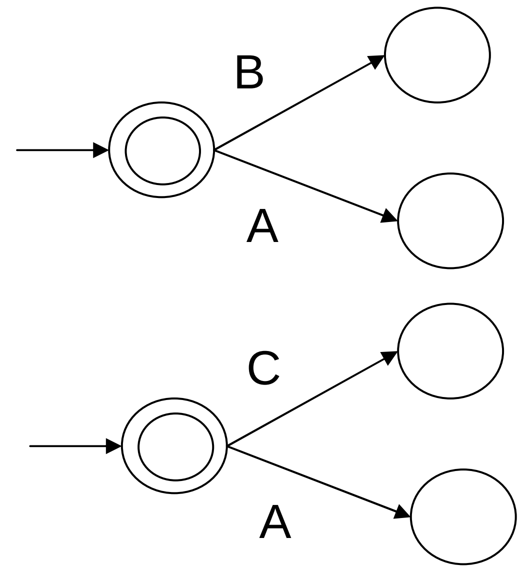
(b) Boolean Abstraction



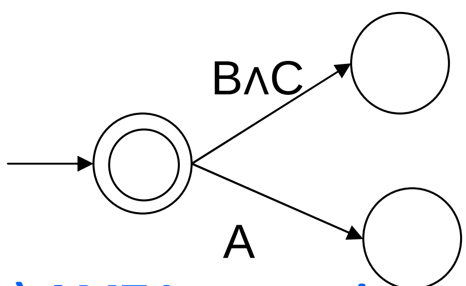
AMT Intersection



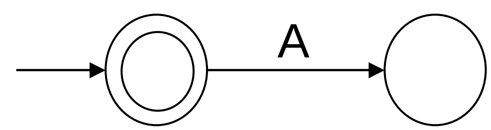
(a) Example of Automata



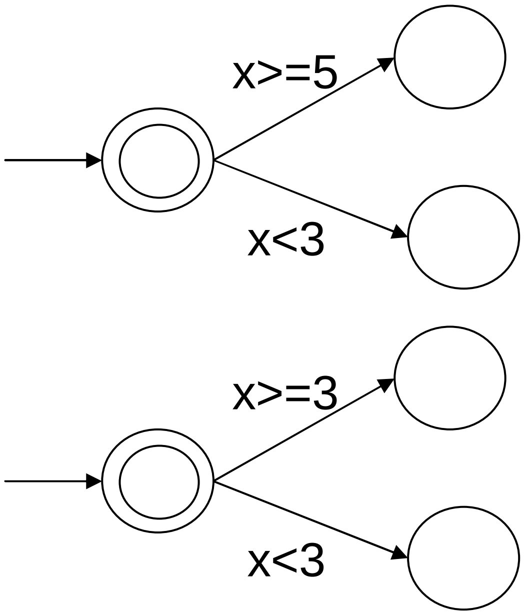
(b) Boolean Abstraction



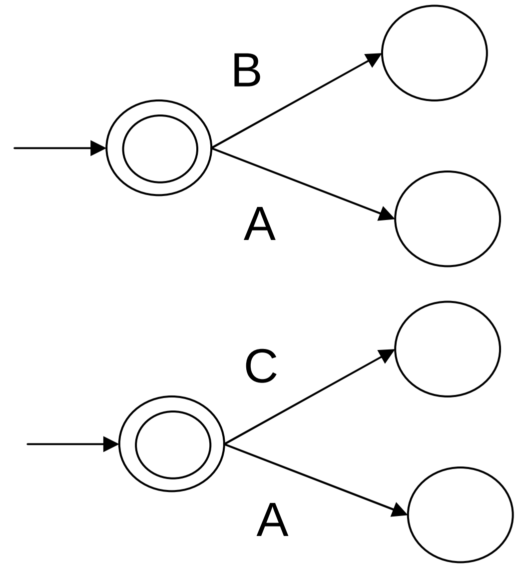
(c) AMT Intersection



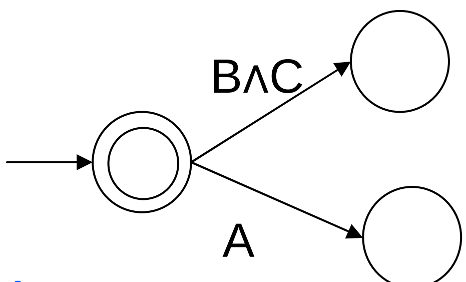
AMT Intersection



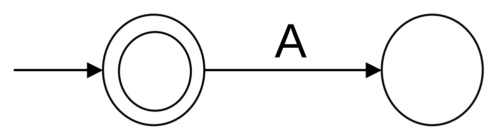
(a) Example of Automata



(b) Boolean Abstraction



(c) AMT Intersection



(d) Normal Intersection

So, What is Contract-Policy Compliance Check ?

- **Security policies as AMT**
- **Matching:**
  - Language Inclusion:
    - Given two automata A^c and A^p representing respectively a contract and a policy, we have a match when the set execution traces of the A^c is a subset of the set of acceptable traces of A^p.
  - Simulation:
    - every security-relevant action invoked by A^c can also be invoked by A^p

Road Map

Security-by-Contract

Automata Modulo Theory

On-the-fly Matching

Simulation Matching

IRM Optimization

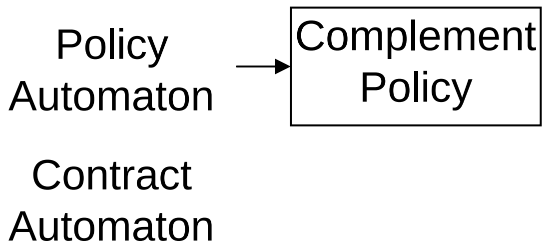
Contract-Policy Matching

Policy
Automaton

Contract
Automaton

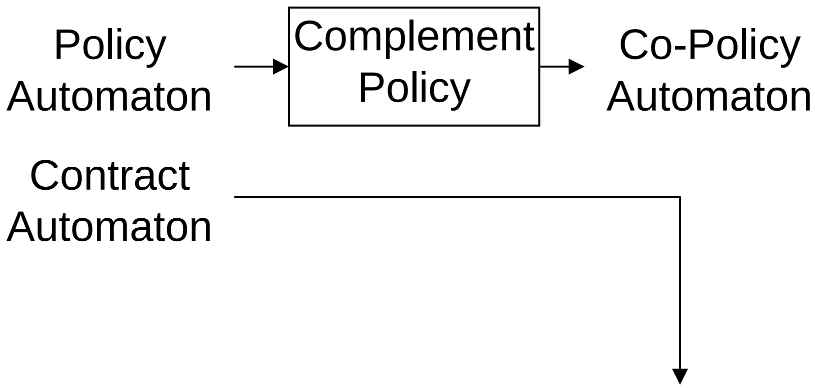
- Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.

Contract-Policy Matching



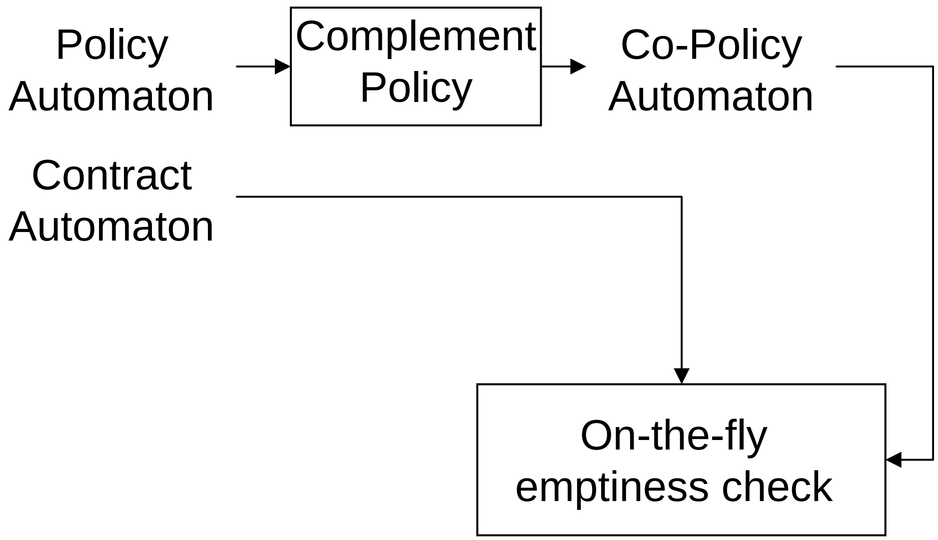
- Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.

Contract-Policy Matching



- Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.

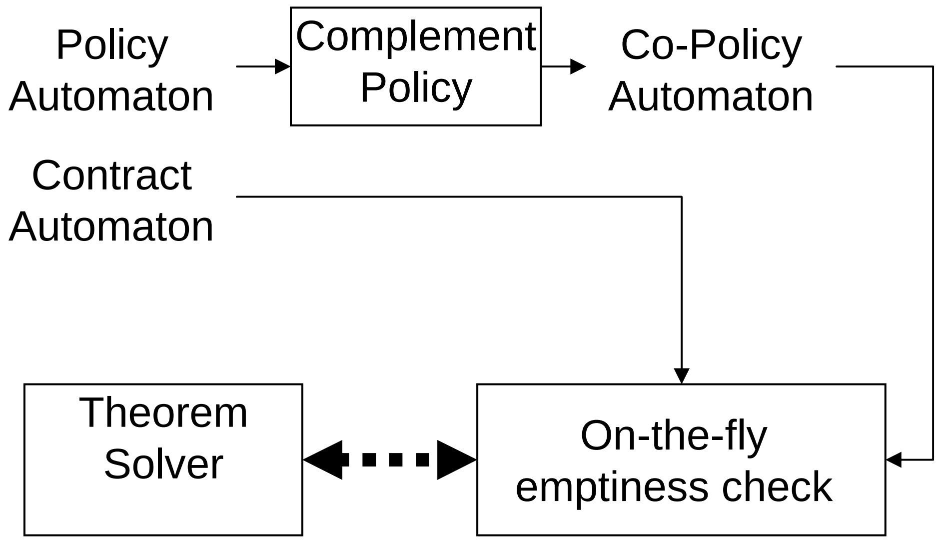
Contract-Policy Matching



- Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.

Università degli Studi di Trento

Contract-Policy Matching



- Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.

Contract-Policy Matching Algorithm

- Input: a contract and a complement policy
- Output: fail or succeed
- Process:
  - **starts a depth first search procedure check_safety from initial state**
  - **IF an accepting state in AMT is reached:**
    - **IF the state contains an error state of complemented policy THEN report a security policy violation without further ado**
    - **IF the state does not contain an error state of complemented policy THEN start a new depth first search check_availability from the candidate state to determine whether it is in a cycle**
    - **IF cycle THEN report an availability violation**

Contract-policy Matching's Result using Language Inclusion

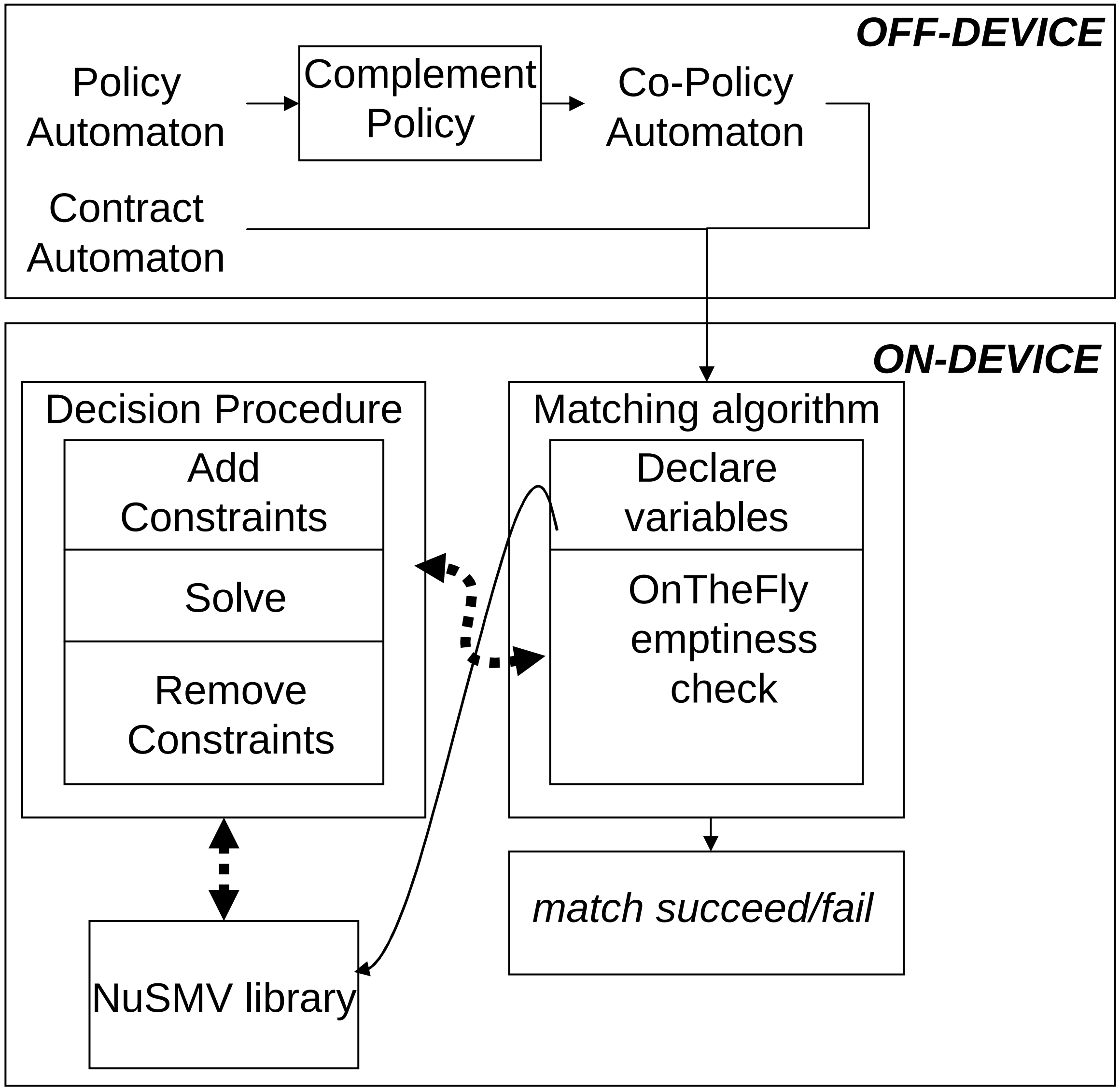
Proposition 4.1.

Let the theory $\mathcal{T}$ be decidable with an oracle for the SMT problem in the complexity class C then:

The contract-policy matching problem for AMT using language inclusion is decidable in

- **time:** $LIN-TIME^{C}$
- **space:** $NLOG-SPACE\text{-}complete^{C}$

Contract-Policy Architecture



Road Map

Security-by-Contract

Automata Modulo Theory

On-the-fly Matching

Simulation Matching

IRM Optimization

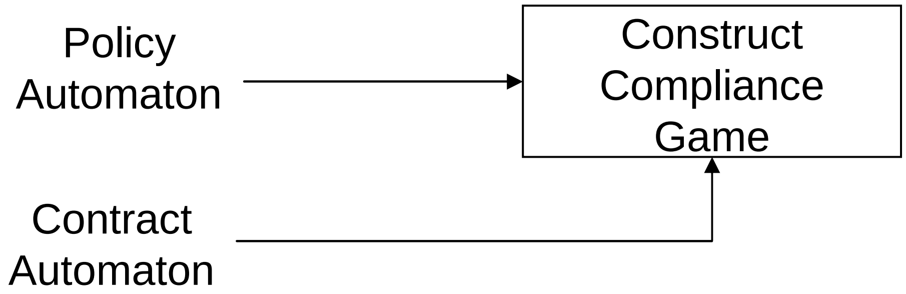
Contract-Policy Matching

Policy
Automaton

Contract
Automaton

- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy
- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  - Adaptation of Jurdzinski's algorithm on parity games (Jurdzinski 2000)

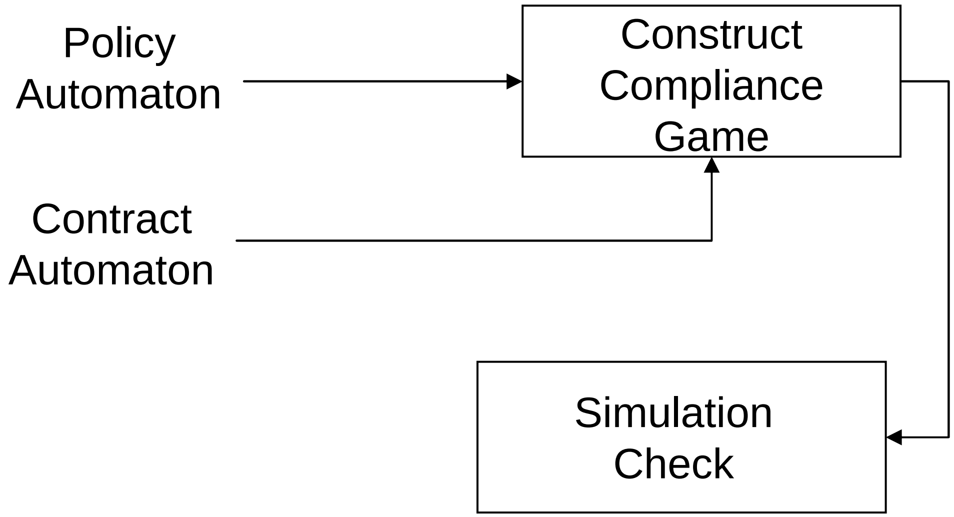
Contract-Policy Matching



- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy
- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  - Adaptation of Jurdzinski's algorithm on parity games (Jurdzinski 2000)

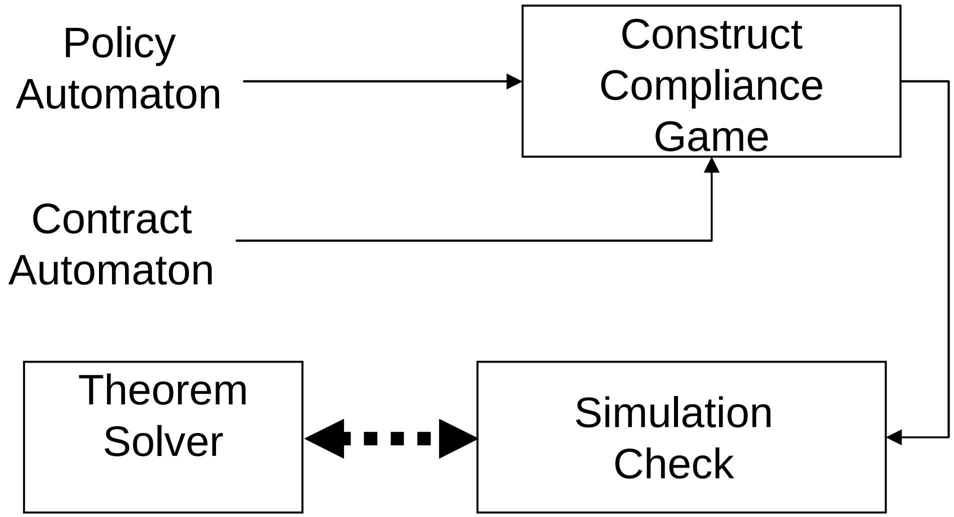
Università degli Studi di Trento

Contract-Policy Matching



- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy
- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  - Adaptation of Jurdzinski's algorithm on parity games (Jurdzinski 2000)

Contract-Policy Matching

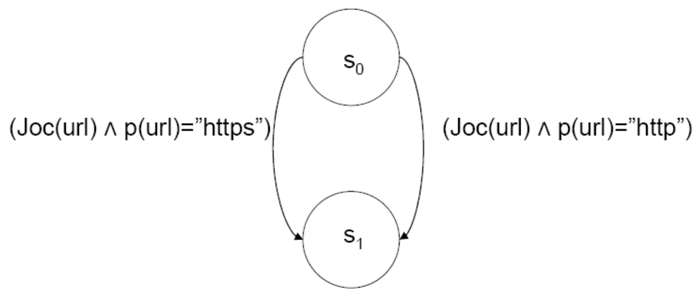


- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy
- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  - Adaptation of Jurdzinski's algorithm on parity games (Jurdzinski 2000)

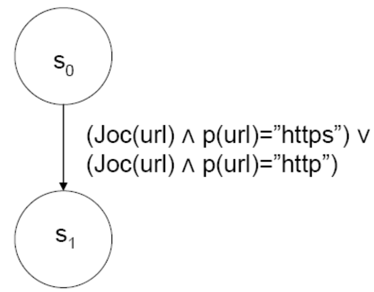
Simulation as Compliance Game

- **Winner of the game:**
  - Contract cannot move: Policy wins.
  - Policy cannot move: Contract wins.
  - Otherwise, two infinite concrete runs *s* and *t* resp. of Contract and Policy:
    - *s* is an accepting concrete run and *t* is not an accepting concrete run: Contract wins.
    - Other cases: Policy wins
- **Failure of Matching**
  - Policy cannot move => Contract is non-compliant

Symbolic vs Concrete Automaton

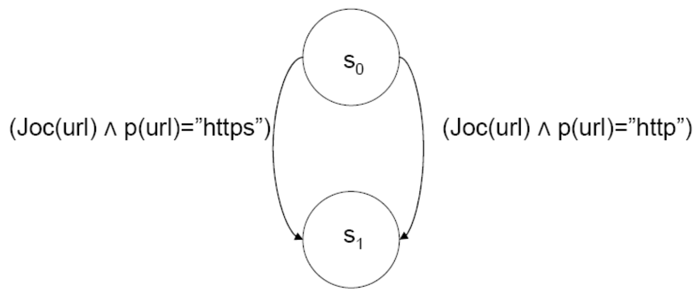


(a) Splitting Edges

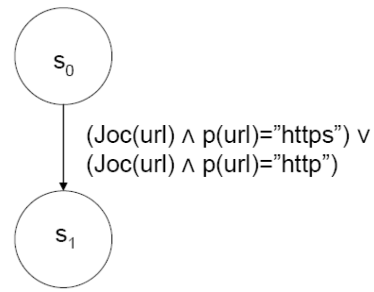


(b) Disjuncting Expressions

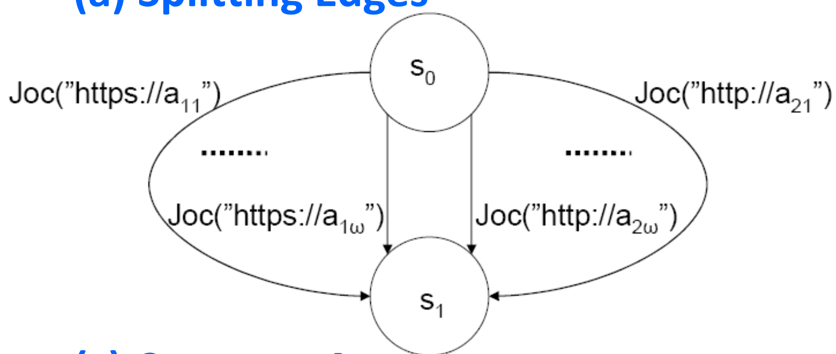
Symbolic vs Concrete Automaton



(a) Splitting Edges



(b) Disjuncting Expressions



(c) Concrete Automaton

$$
\begin{aligned}
e_{11} &\doteq (Joc(url) \wedge p(url) = \text{“https”}) \\
e_{12} &\doteq (Joc(url) \wedge p(url) = \text{“http”}) \\
e_{2} &\doteq (Joc(url) \wedge p(url) = \text{“https”}) \\
&\quad \vee (Joc(url) \wedge p(url) = \text{“http”})
\end{aligned}
$$

(d) Abbreviations

- **IF A^c *complies* with A^p THEN A^c *concretely complies* with A^p**
  - The converse does not hold in general.
  - Contrast to the simulation notions of (Hennessy and Lin 1995)
- **AMT fair simulation is stronger than AMT language inclusion**

Normalized AMT

- **For every q, q_1 in set of states S there is at most one expression e_1 in set of expressions $\mathscr{E}$ s.t. (q, e_1, q_1) is in set of transitions Δ**
  - Example: from previous figure (a) is NOT normalized, (b) is normalized
- **Normalization is possible when:**
  - theory $\mathcal{T}$ is convex and closed under disjunction.
- **Normalization preserves AMT determinism**
- **For normalized AMT: A^c *concretely complies* with A^p IFF A^c *complies* with A^p**

Simulation Policy-Contract Algorithm

- Matching between a contract with a security policy problem can be reduced to compliance game between a contract with a policy.
- Input: a contract and a policy
- Output: fail or succeed
- Process:
  - **Create compliance game graph $G = \langle V, E, l \rangle$**
  - **$\mu(v) := 0$ for all $v \in V$**
  - **WHILE $\mu(v) \neq \mu_{new}(\mu, v)$ for some $v \in V$ DO**
    - $\mu := \mu_{new}(\mu, v)$
  - **IF $\mu(v(s_0^c, s_0^p)) < \infty$ THEN**
    - succeed (Simulation exists)

Contract-policy Matching's Result using Simulation

Proposition 6.2.

Let the theory $\mathcal{T}$ be decidable with an oracle for the SMT problem in the complexity class *C* then:

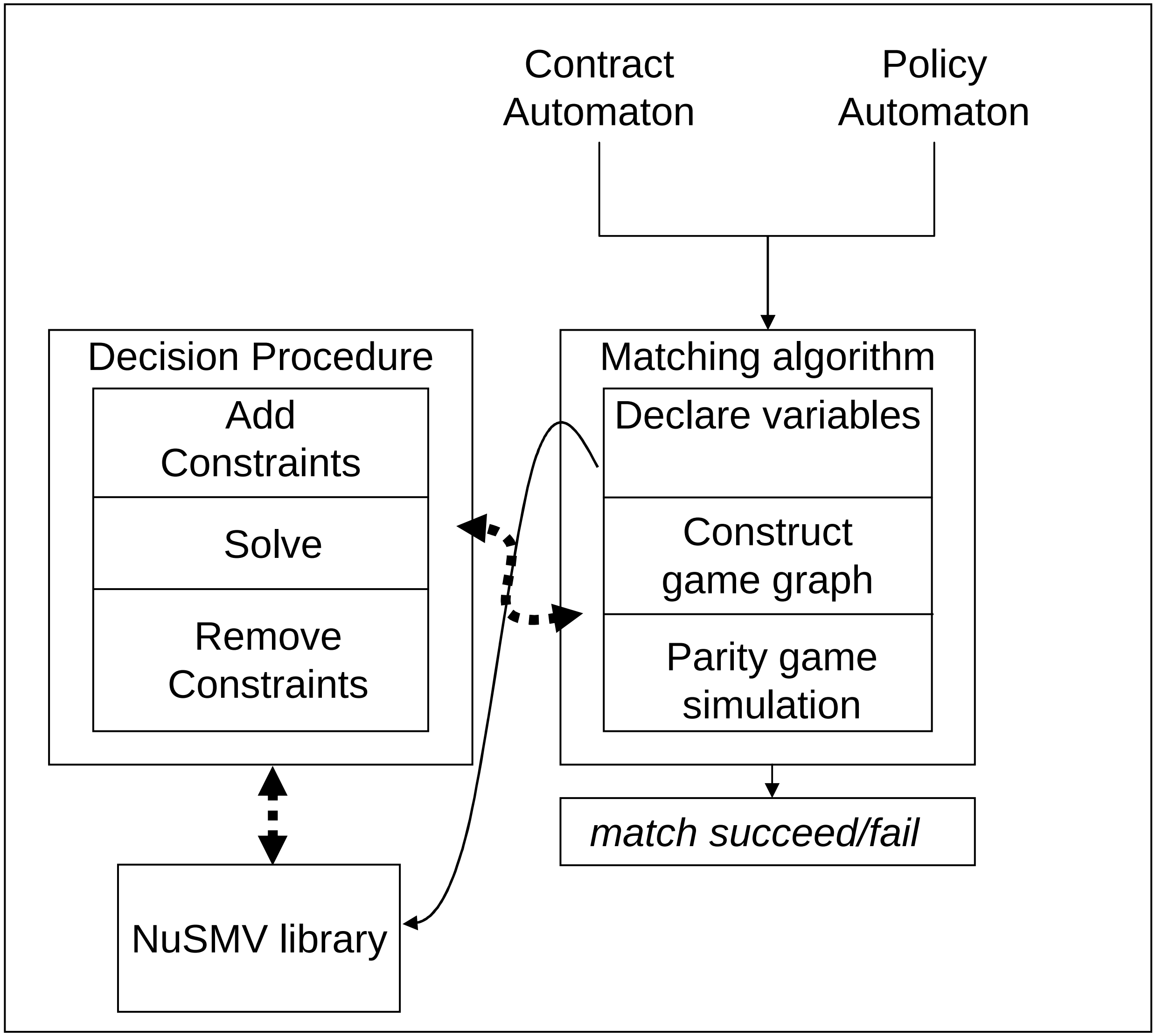
The contract-policy matching problem for AMT using fair simulation is decidable in

- **time: *O(2. |E| .|V1|)***
- **space: *O(|V|)***

– **By Lemma 6.1.**

- **$|V_1|$ is in $O(|S^c| \,.\, |S^p|)$**
- **$|V_0|$ is in $O(|S^c| \,.\, |S^p| \,.\, |\Delta^c|)$**
- **$|E|$ is in $O(|S^c| \,.\, |S^p| \,.\, |\Delta^c|)^C$**

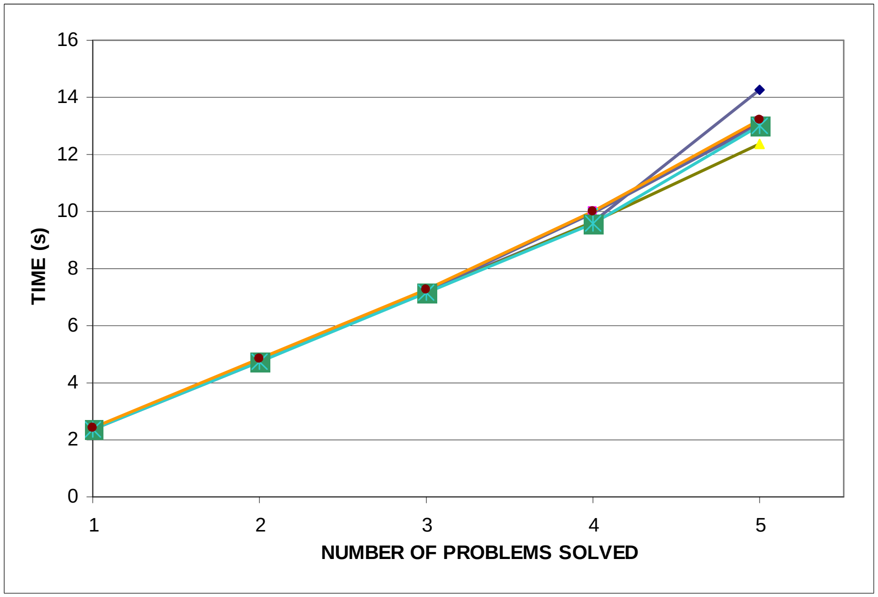
Simulation Contract-Policy Architecture



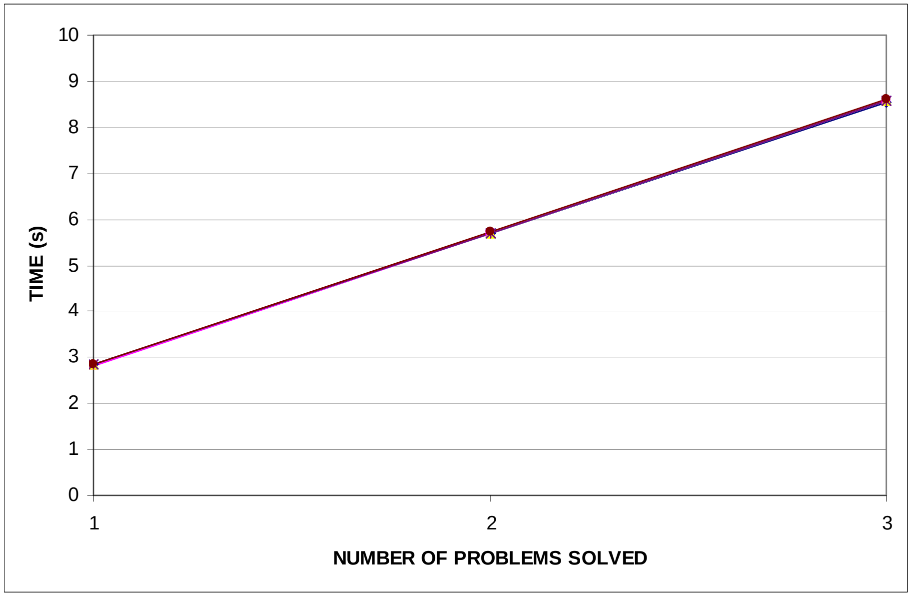
Matching Experiment

- Goal: proof-of-concept and deciding the best configuration of integrating matching algorithm with decision procedure
- Collected data: number of visited states, number of visited transitions, and running time for each problem in each design alternative
- Problem suite:
  - **sample of policy-contract (mis)matching pairs**
  - **artificial problem to mimic large number of states**
- Setup:
  - **Desktop:**
    - **PC (Intel(R) Pentium D CPU 3.40GHz, 3389.442MHz, 1.99GB of RAM, 2048 KB cache)**
    - **On-the-fly: OS Linux version 2.6.20-16-generic, Kubuntu 7.04 (Feisty Fawn)**
    - **Simulation: Microsoft(R) Windows XP Professional Version 2002 Service Pack 3**
  - **Mobile device:**
    - **HTC P3600 (3G PDA phone) with ROM 128MB, RAM 64MB, 400MHz, Samsung(R) SC32442A**
    - **OS Microsoft(R) Windows Mobile 5.0 with Direct Push technology**

On-the-fly Matching Experiment on Desktop

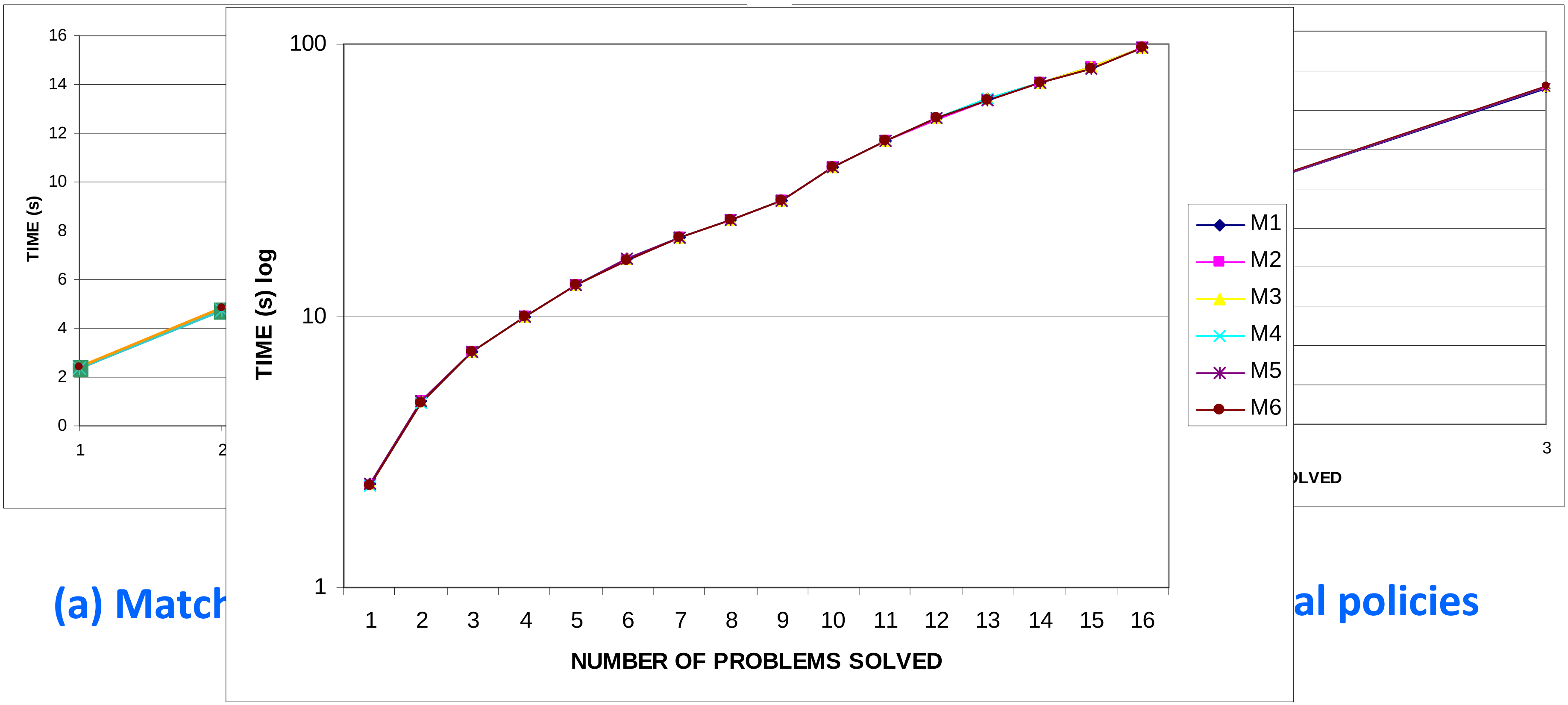


(a) Match succeeds for real policies



(b) Match fails for real policies

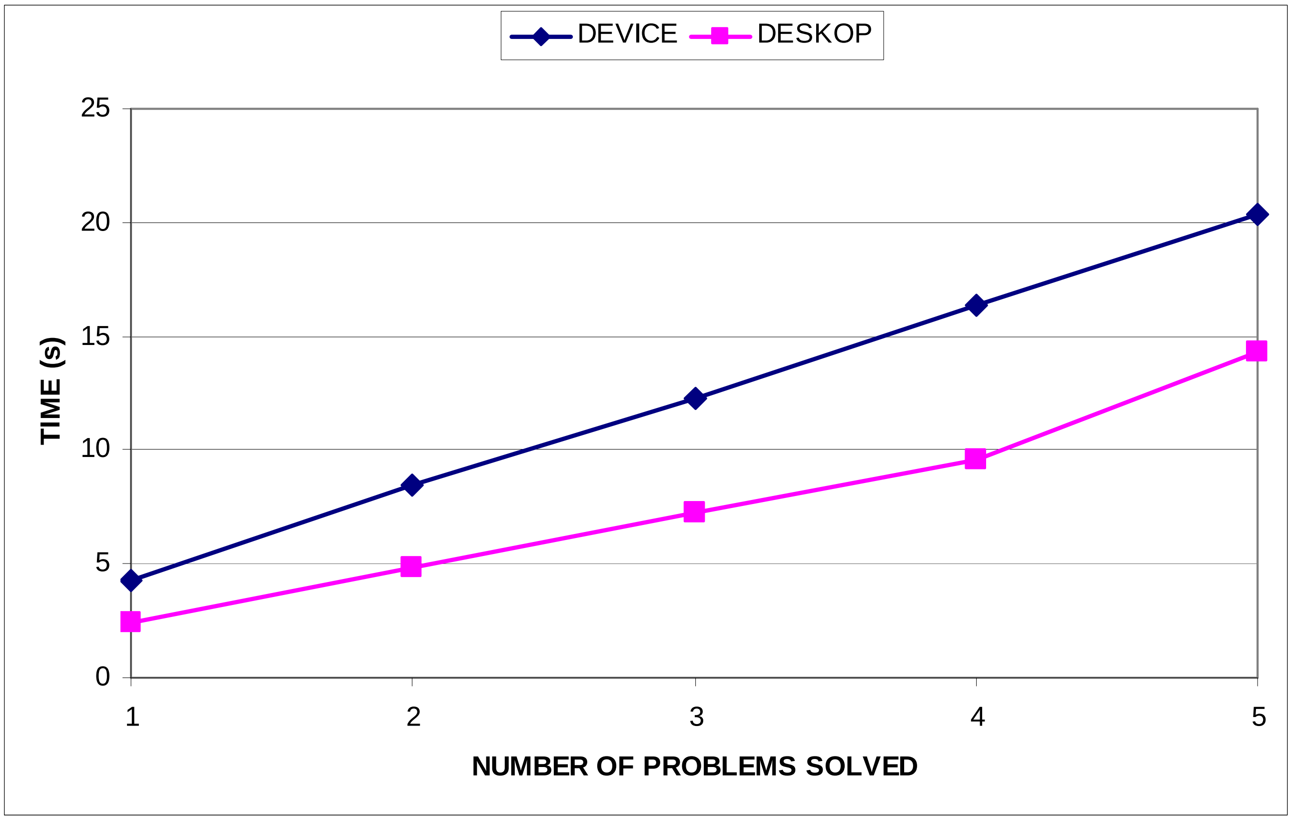
On-the-fly Matching Experiment on Desktop



(a) Match ... al policies

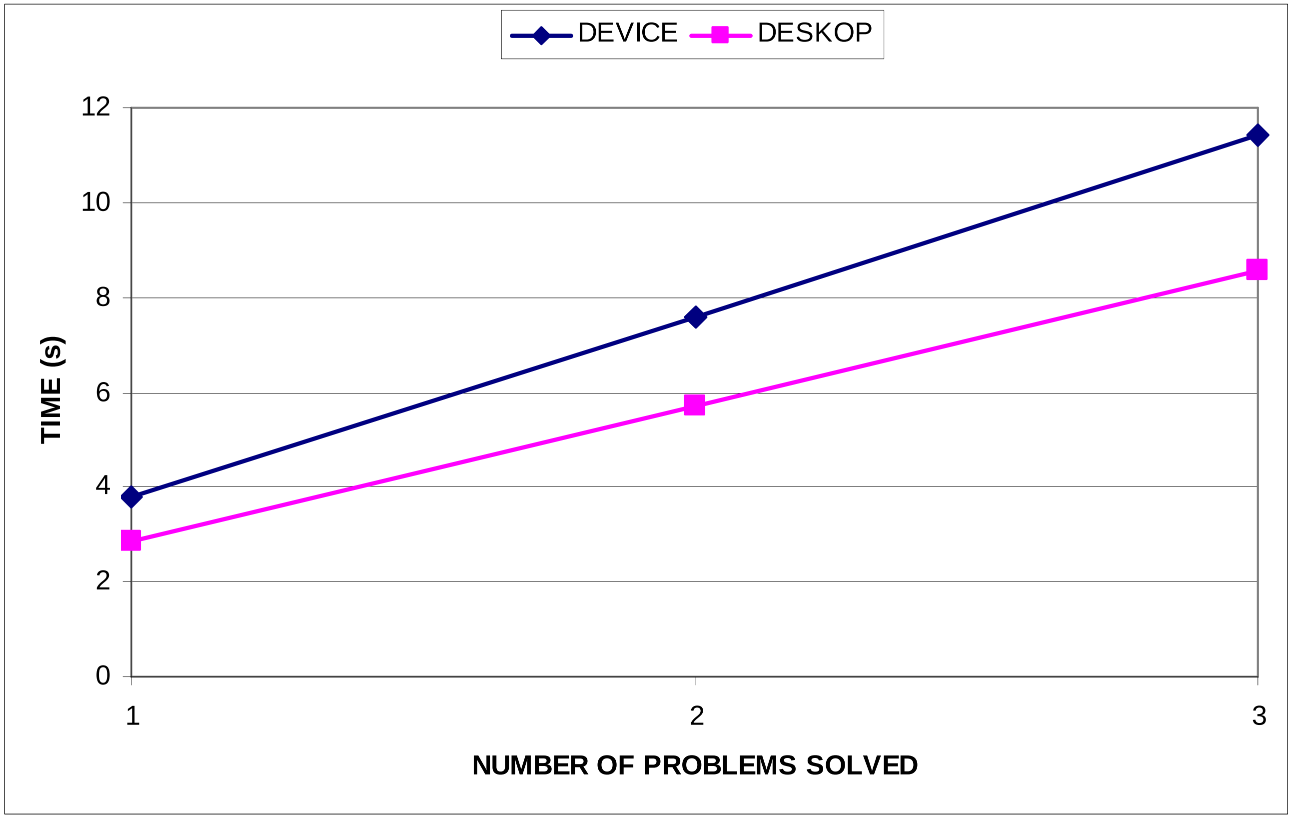
(c) Matches among synthetic contracts and policies

On-the-fly Matching Experiment Device vs Desktop



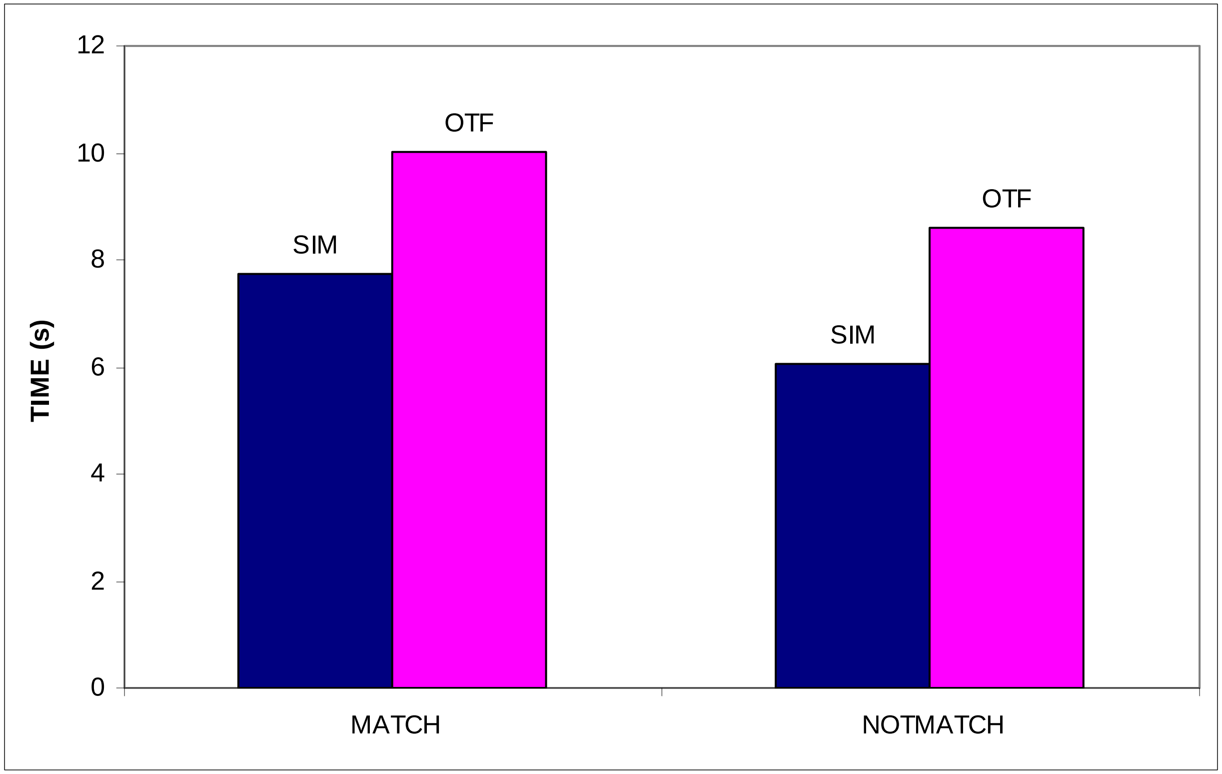
(a) Match succeeds

On-the-fly Matching Experiment Device vs Desktop



(b) Match fails

Matching Experiment
Simulation vs On-the-fly on Desktop



(a) Match succeeds

#SOLVED	SIM (s)	OTF (s)
1	2.014	2.41
2	3.948	4.825
3	5.834	7.263
4	7.72	10.023

(b) Match fails

#SOLVED	SIM (s)	OTF (s)
1	1.998	2.858
2	4.058	5.728
3	6.056	8.602

Road Map

Security-by-Contract

Automata Modulo Theory

On-the-fly Matching

Simulation Matching

IRM Optimization

IRM Optimization Models

Trusted | **Untrusted**

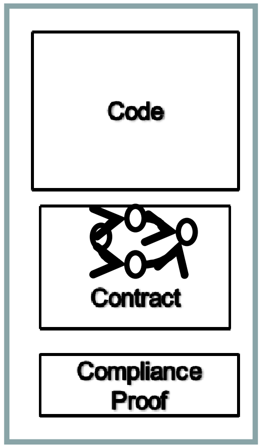
extract security relevant behaviors from code

ContractExtractor

Code

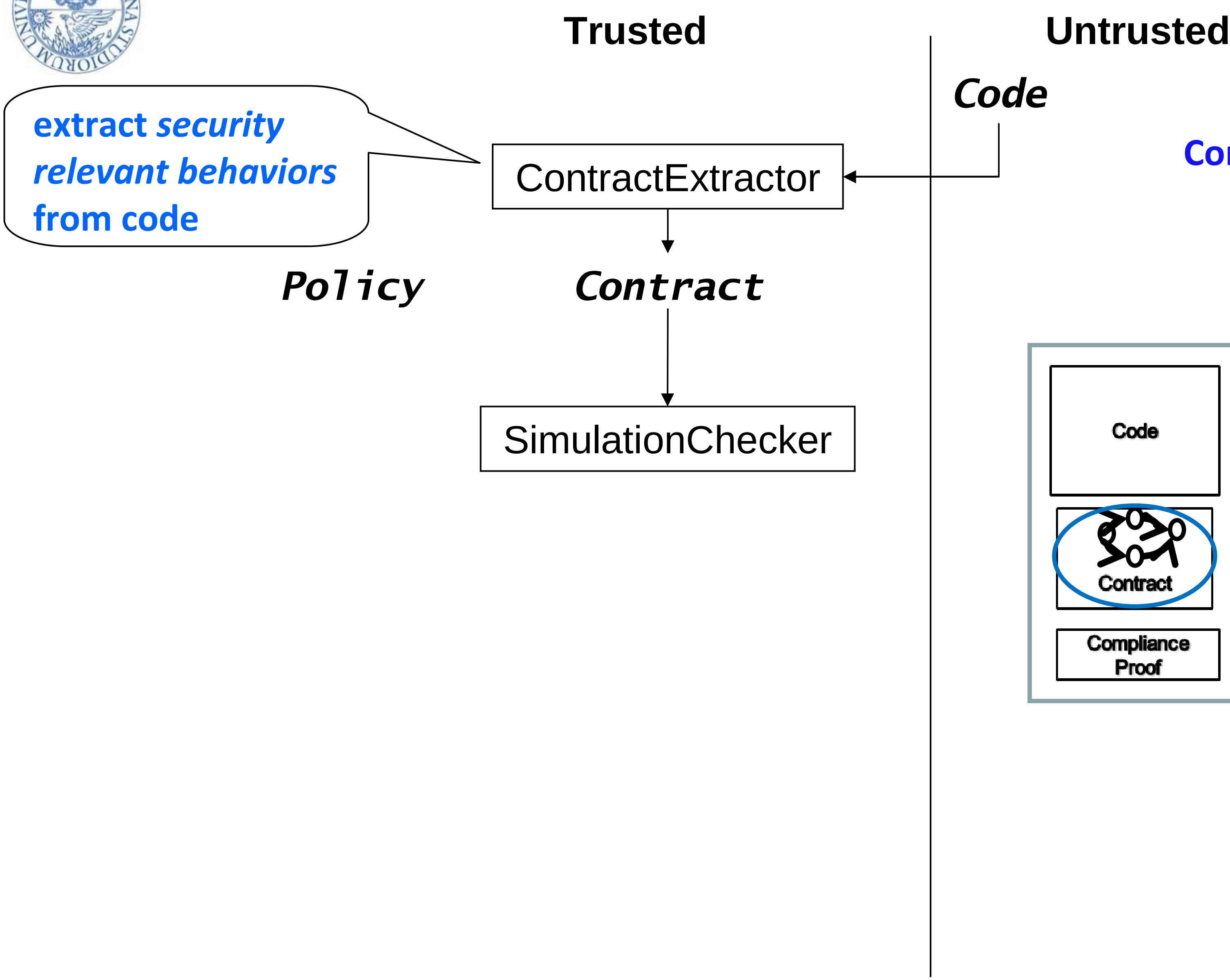
Contract

Model1: Contract Extractor on Trusted part



Università degli Studi di Trento

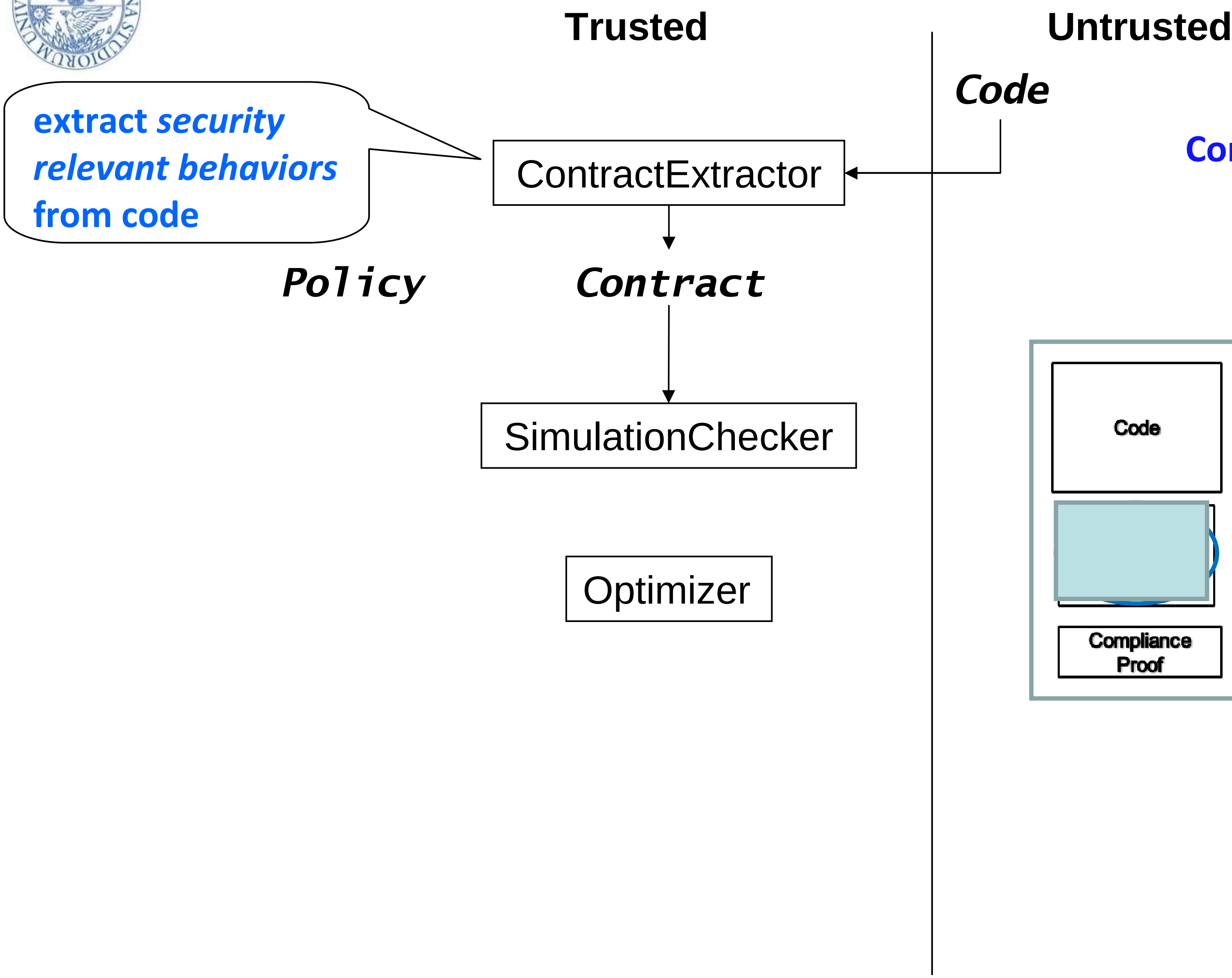
IRM Optimization Models



Model1: Contract Extractor on Trusted part

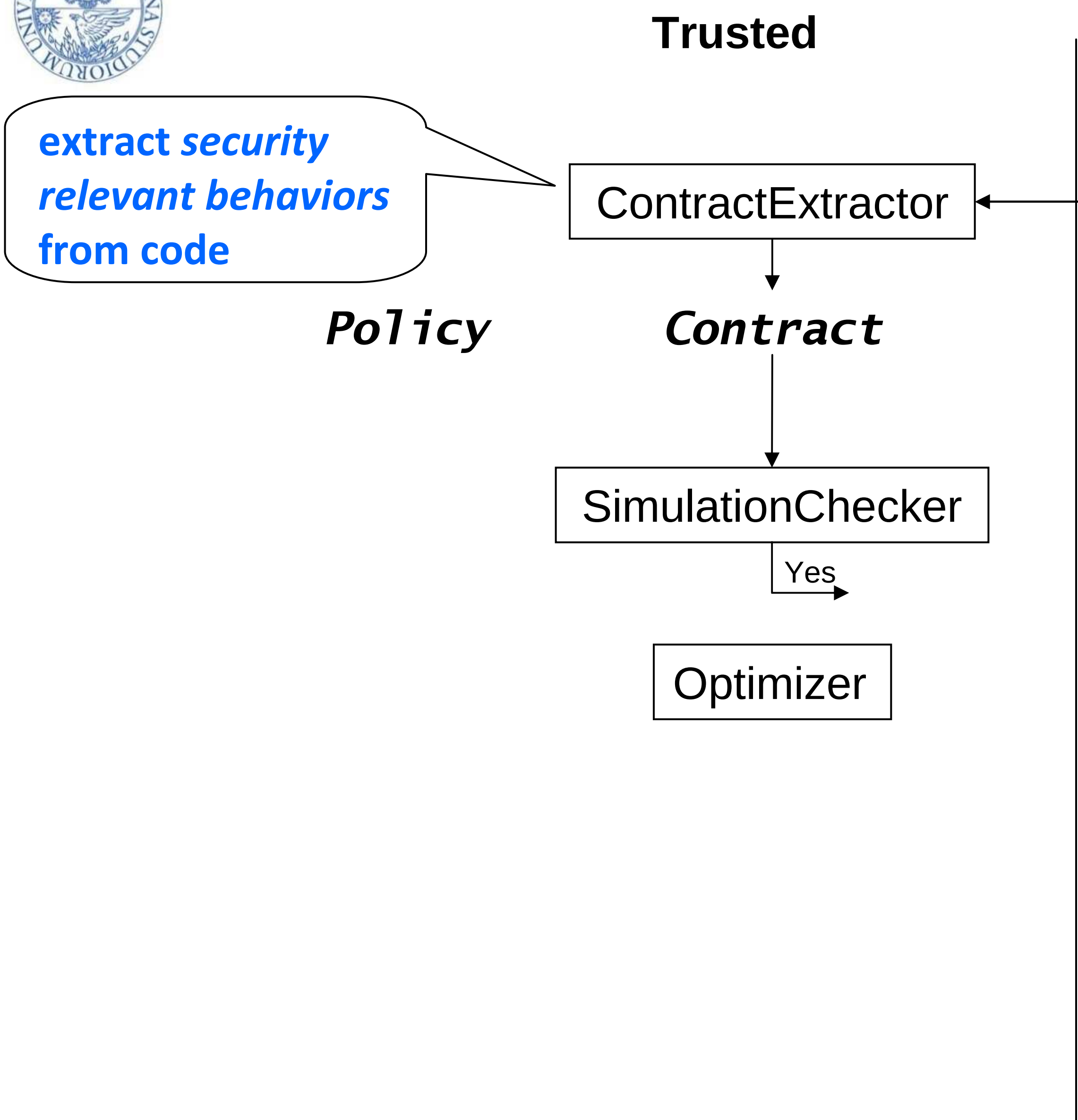
Università degli Studi di Trento

IRM Optimization Models

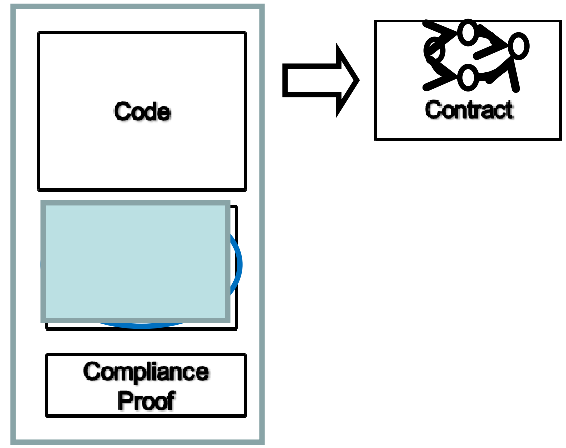


Model1: Contract Extractor on Trusted part

IRM Optimization Models

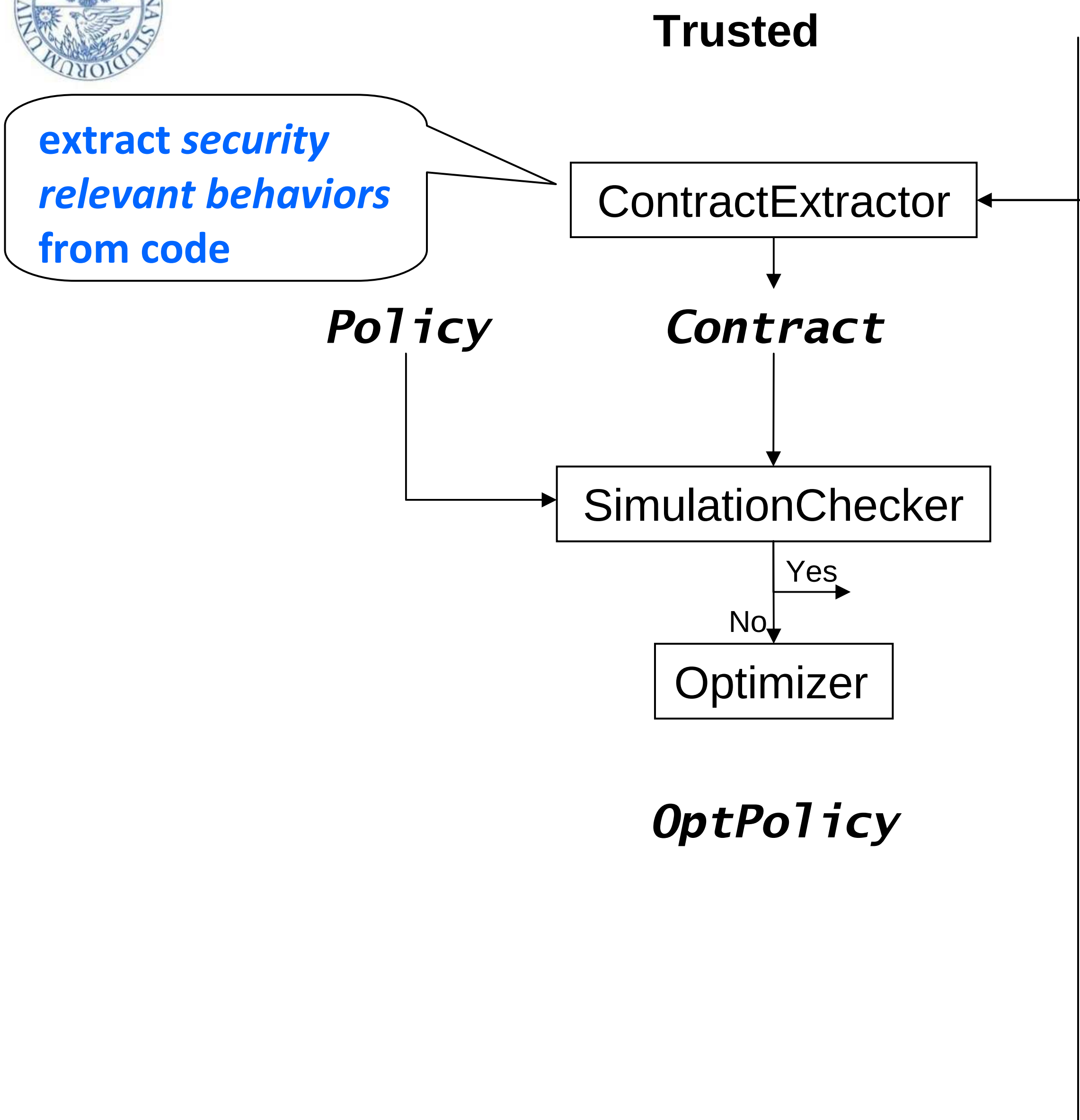


Model1:
Contract Extractor
on Trusted part

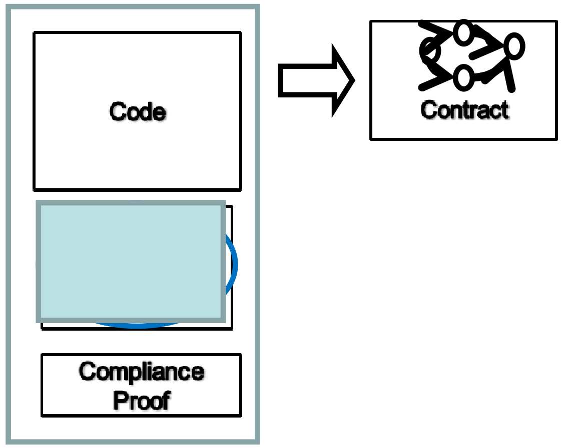


Università degli Studi di Trento

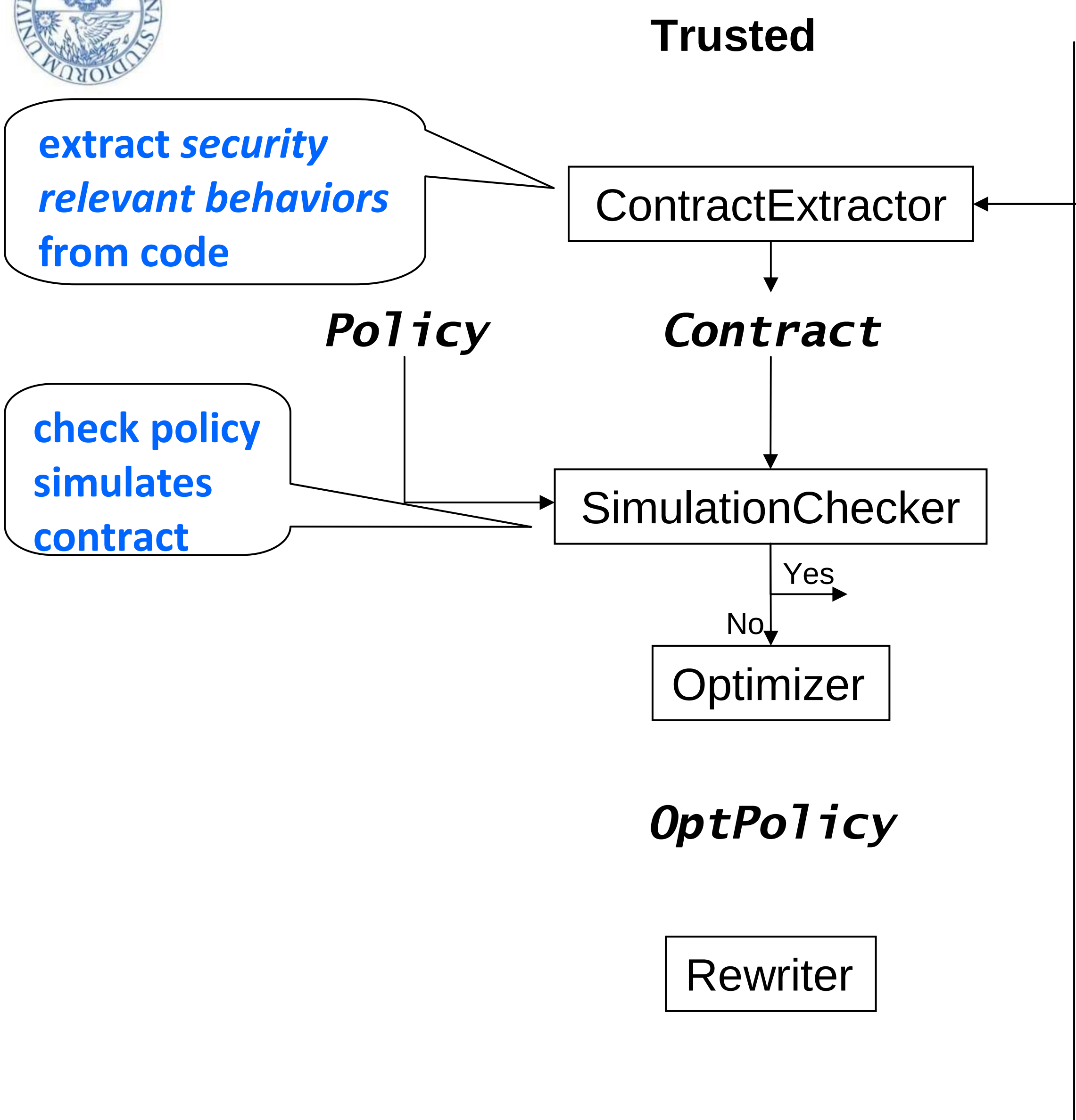
IRM Optimization Models



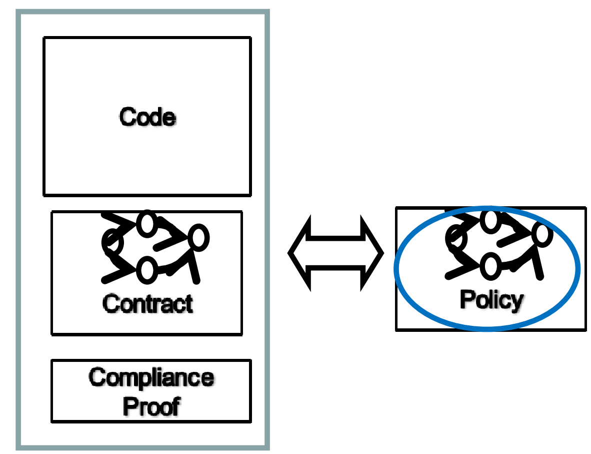
Model1: Contract Extractor on Trusted part



IRM Optimization Models

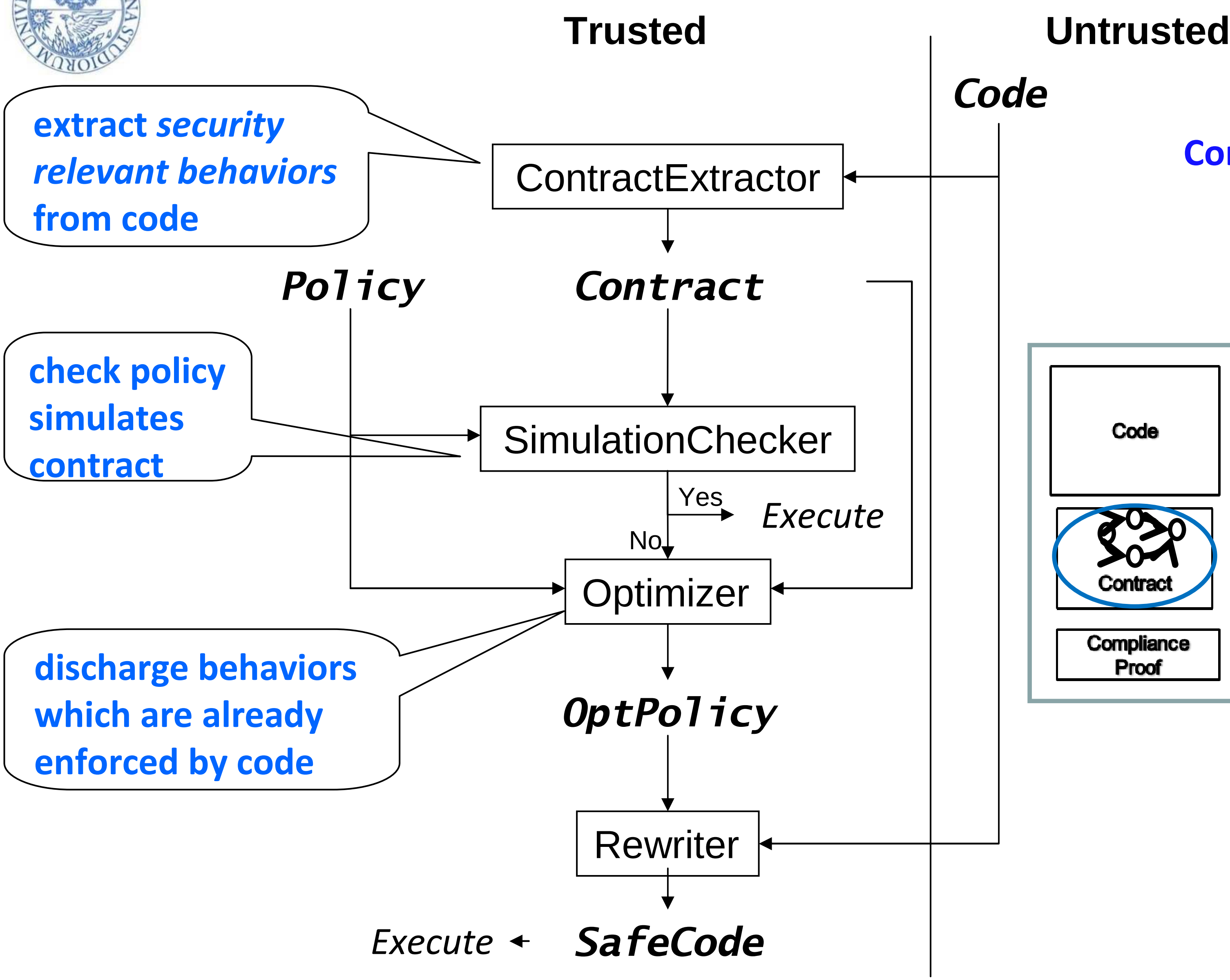


Model1:
Contract Extractor on Trusted part

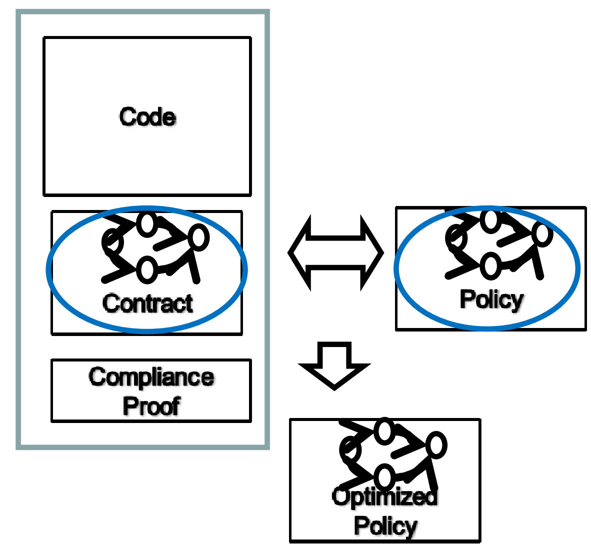


Università degli Studi di Trento

IRM Optimization Models

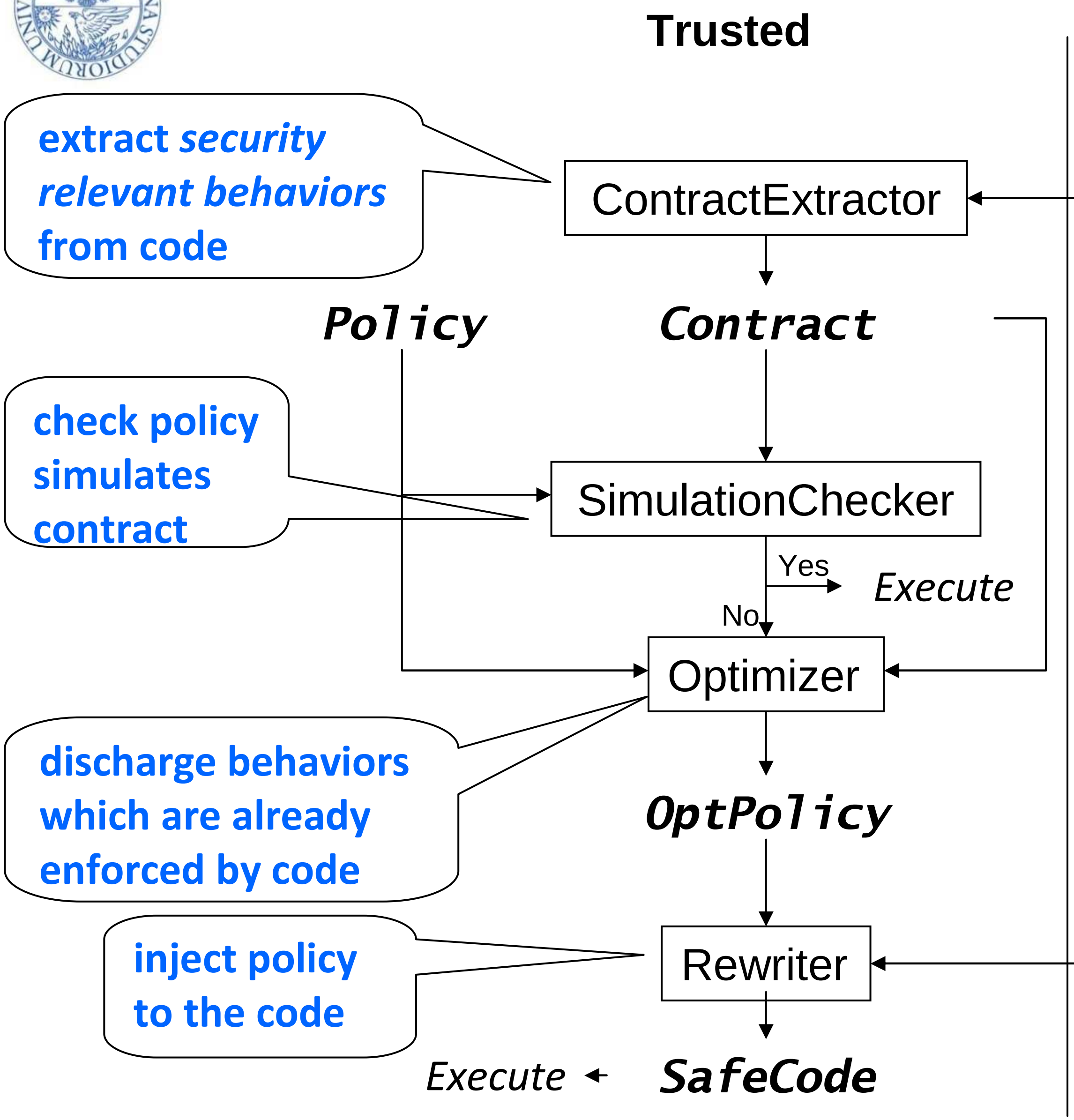


Model1: Contract Extractor on Trusted part

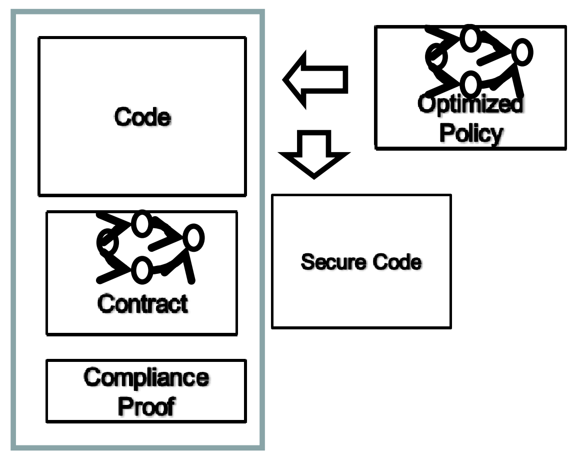


Università degli Studi di Trento

IRM Optimization Models



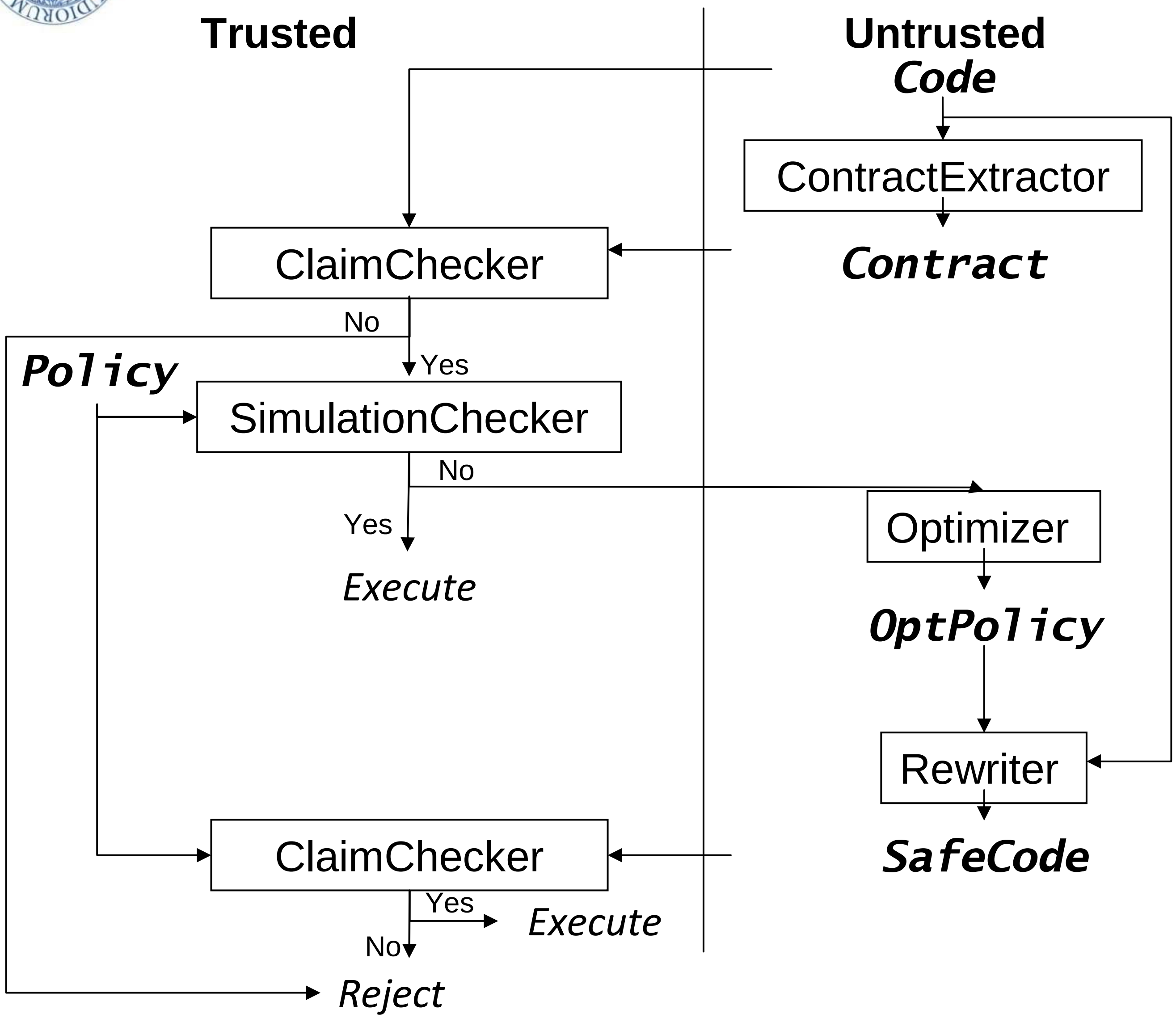
Model1:
Contract Extractor
on Trusted part



Università degli Studi di Trento

Optimizer and Rewriter on Untrusted part

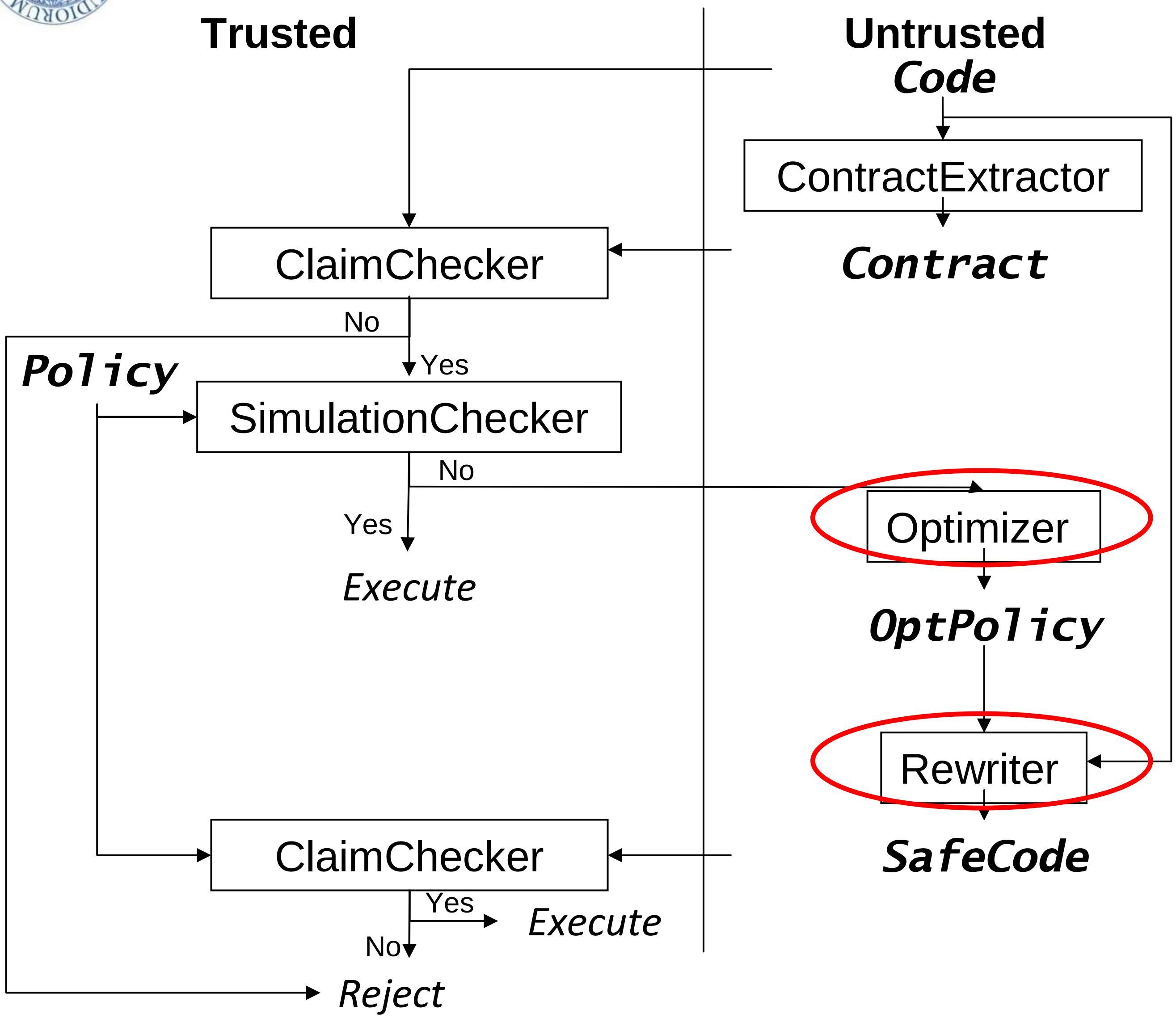
Model6: Contract Extractor on Untrusted part



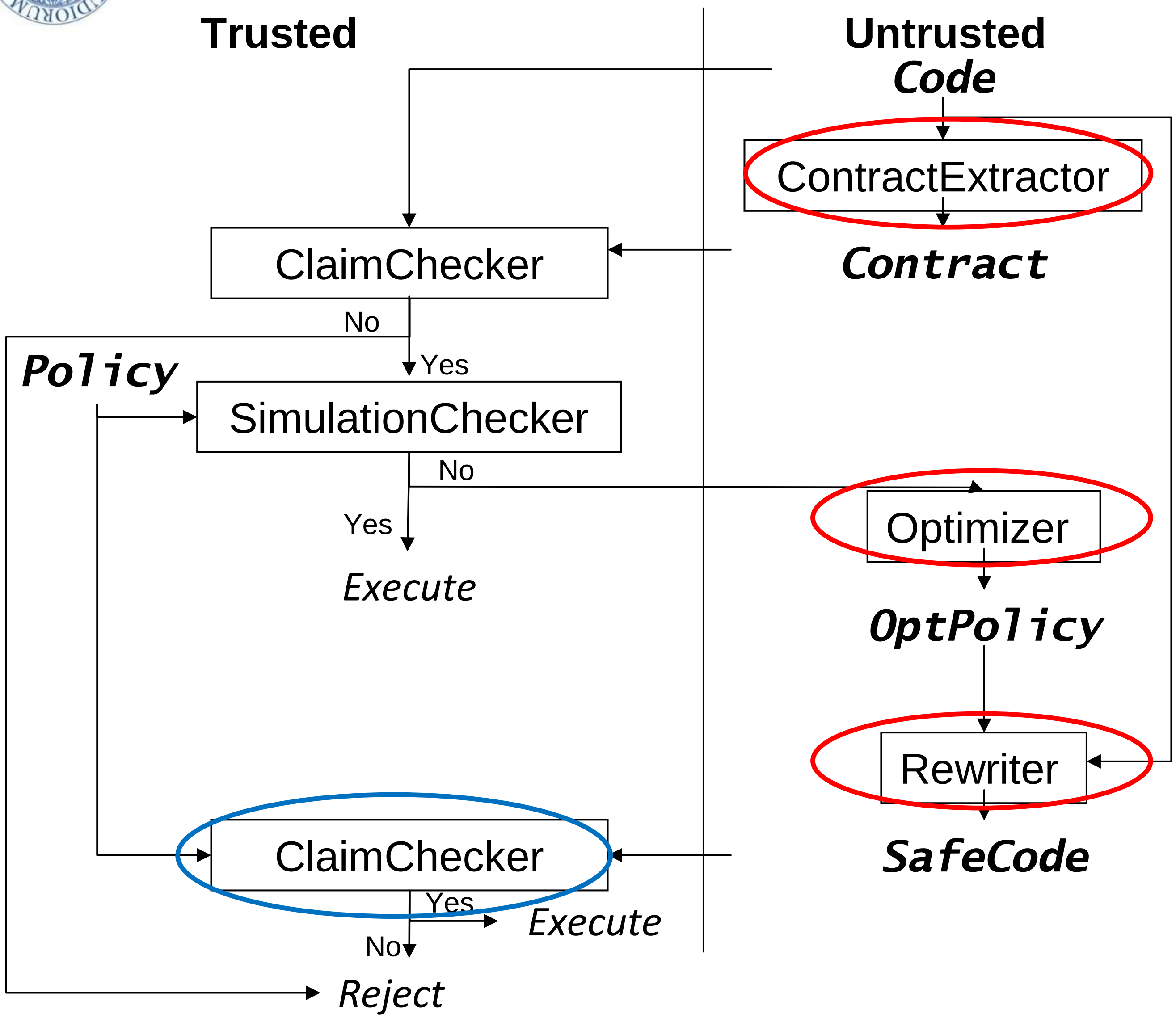
Università degli Studi di Trento

Optimizer and Rewriter on Untrusted part

Model6: Contract Extractor on Untrusted part



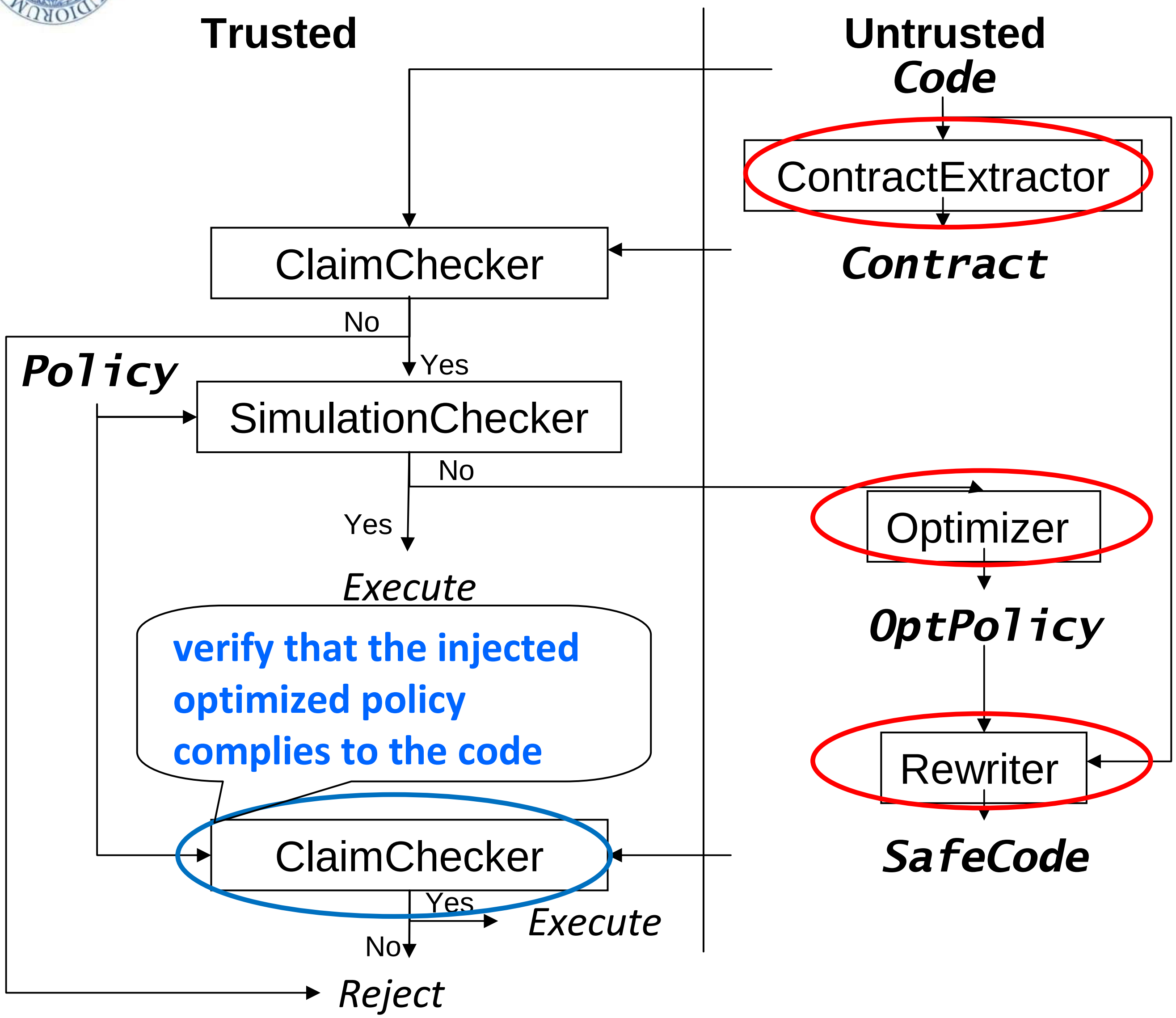
Optimizer and Rewriter on Untrusted part



Model6:
Contract Extractor on Untrusted part

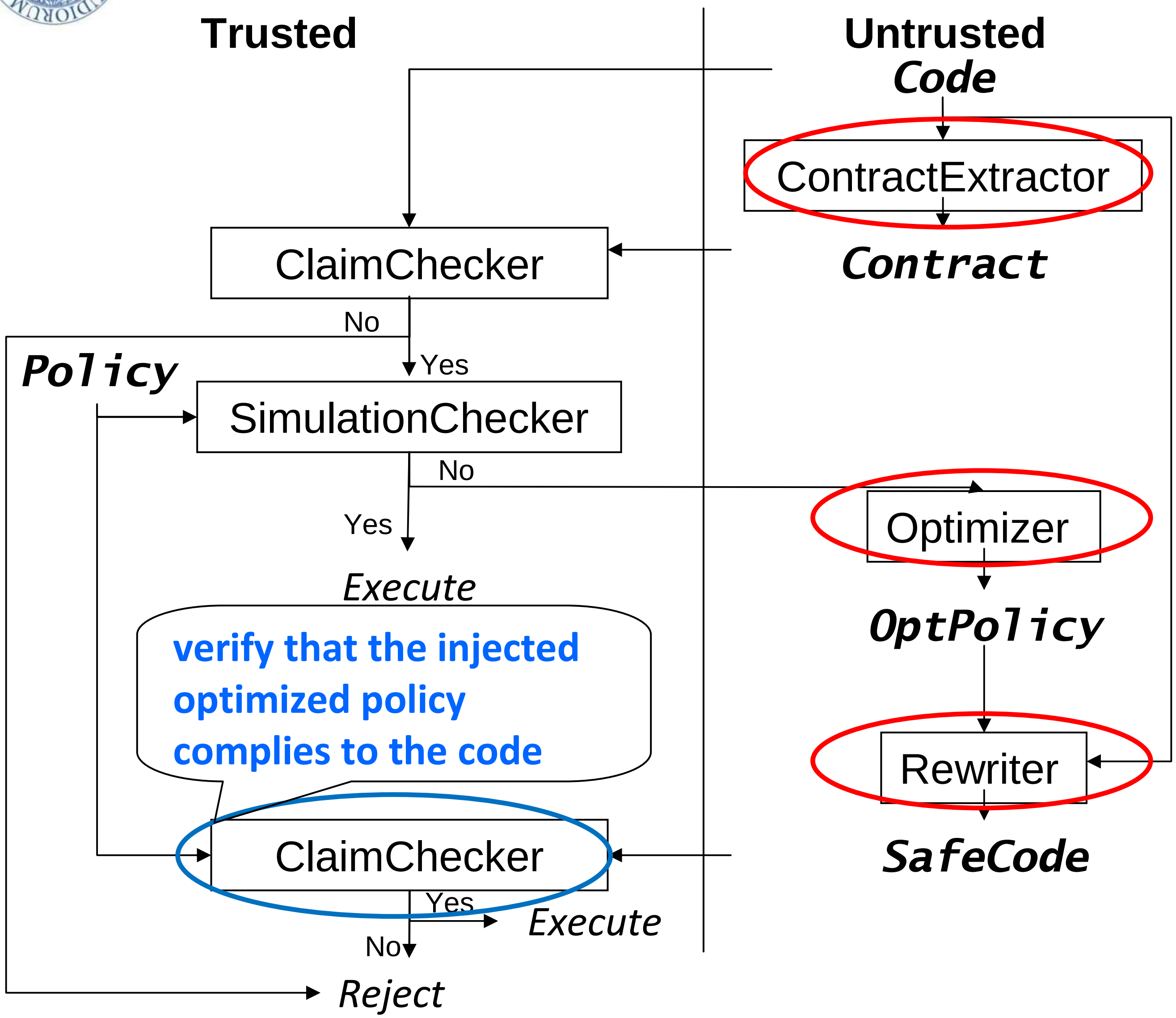
Optimizer and Rewriter on Untrusted part

Model6: Contract Extractor on Untrusted part

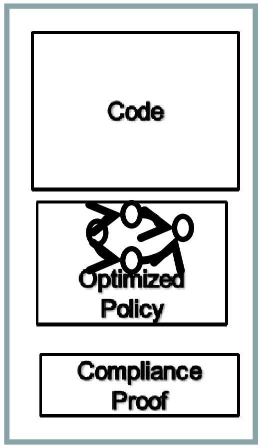


Università degli Studi di Trento

Optimizer and Rewriter on Untrusted part



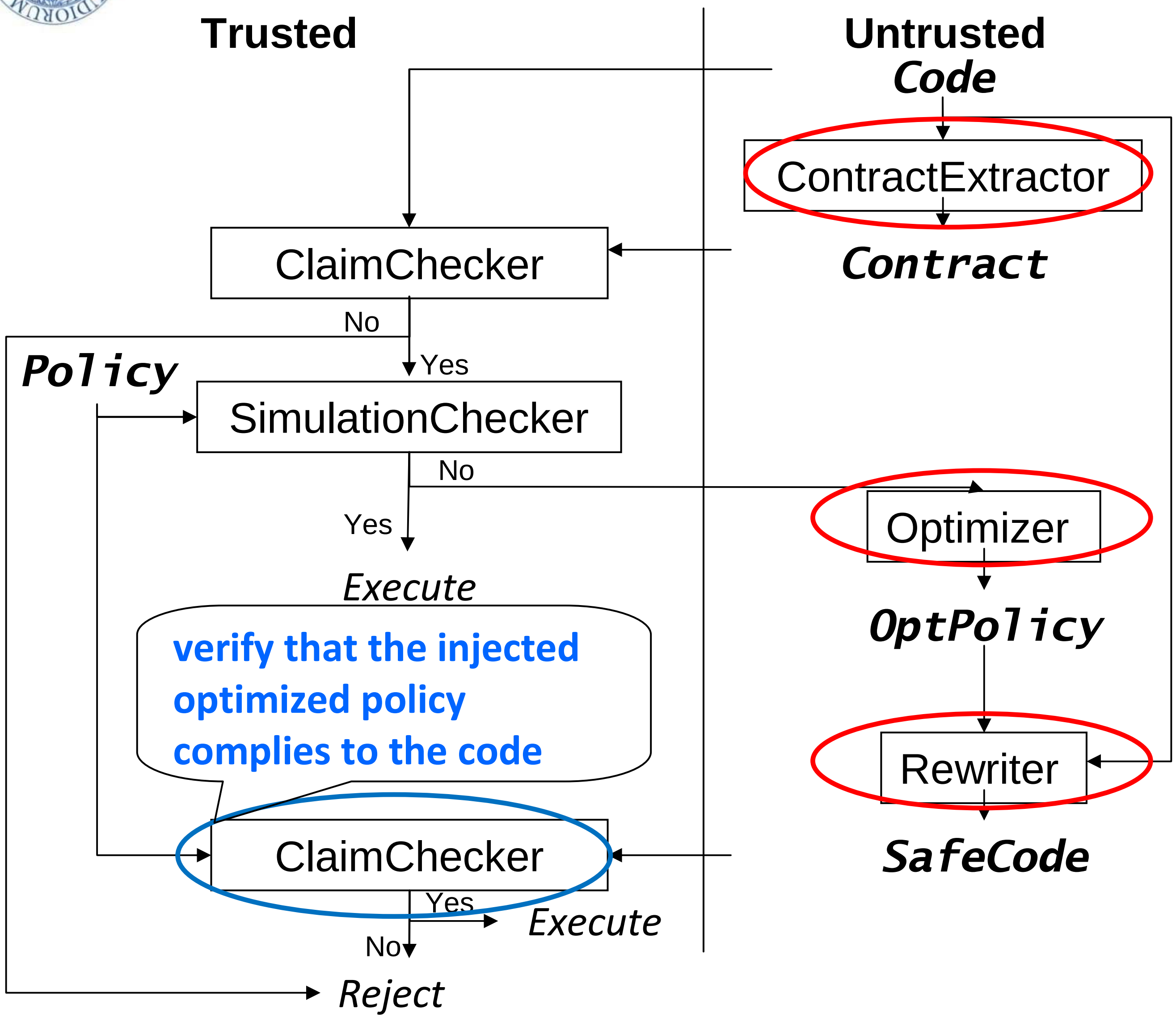
Model6: Contract Extractor on Untrusted part

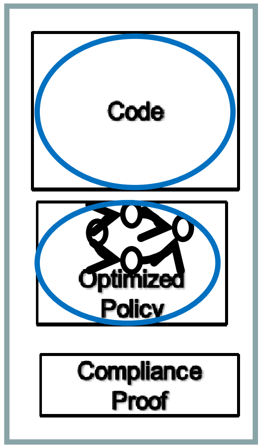


Università degli Studi di Trento

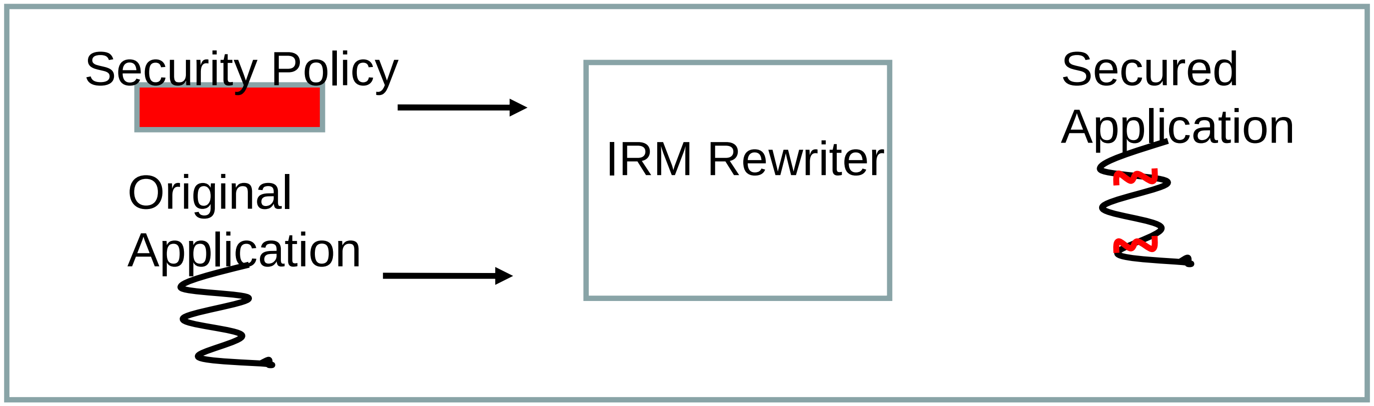
Optimizer and Rewriter on Untrusted part

Model6: Contract Extractor on Untrusted part

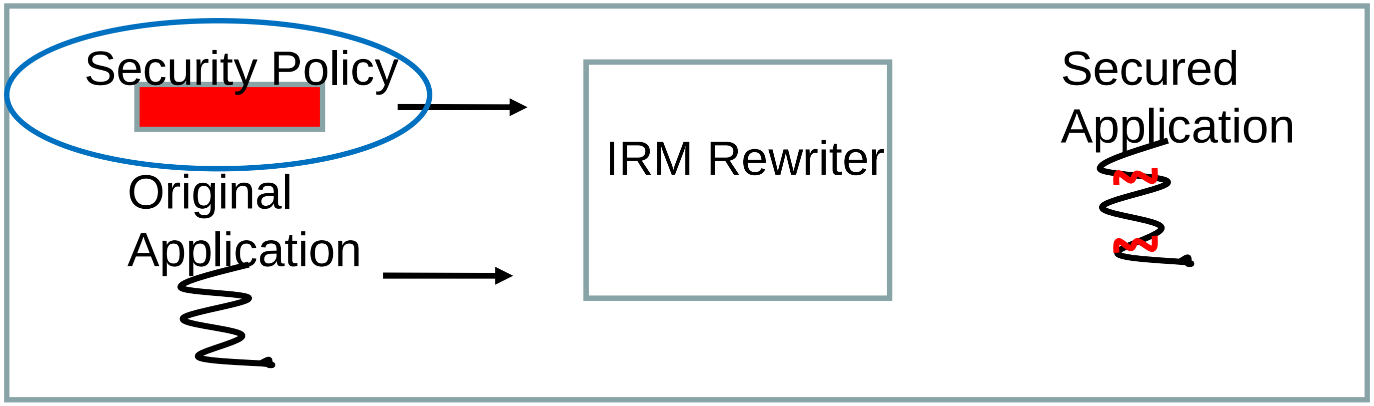




Optimizing Security Policy or Rewriter



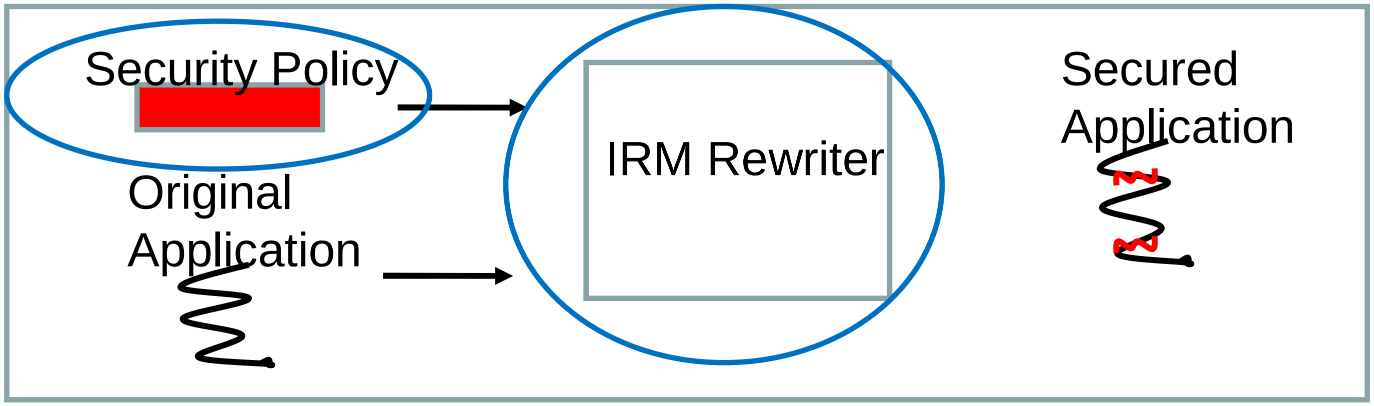
Optimizing Security Policy or Rewriter



- **Security Automata SFI Implementation (SASI) [Erlingson-etal-NSPW’99]**
  - **Minimizing TCB by working at the level of object code**
- **Trade off between moving more processes out of trusted part and the complexity of the whole process [Hamlen-Thesis’06]**
- **Efficient IRM Enforcement [Yan-etal-ASIACCS’09]**
  - **a constrained representation of history-based access control policies**
  - **exploit the structure of this policy representation**
  - **extended into a distributed optimization protocol**

Università degli Studi di Trento

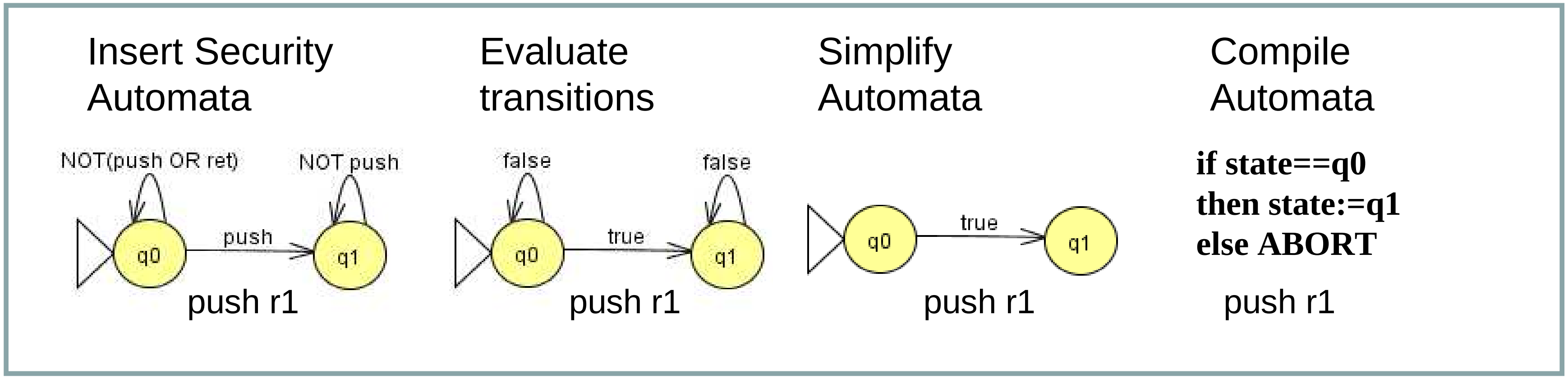
Optimizing Security Policy or Rewriter



- **Security Automata SFI Implementation (SASI) [Erlingson-etal-NSPW’99]**
  - **Minimizing TCB by working at the level of object code**
- **Trade off between moving more processes out of trusted part and the complexity of the whole process [Hamlen-Thesis’06]**
- **Efficient IRM Enforcement [Yan-etal-ASIACCS’09]**
  - **a constrained representation of history-based access control policies**
  - **exploit the structure of this policy representation**
  - **extended into a distributed optimization protocol**

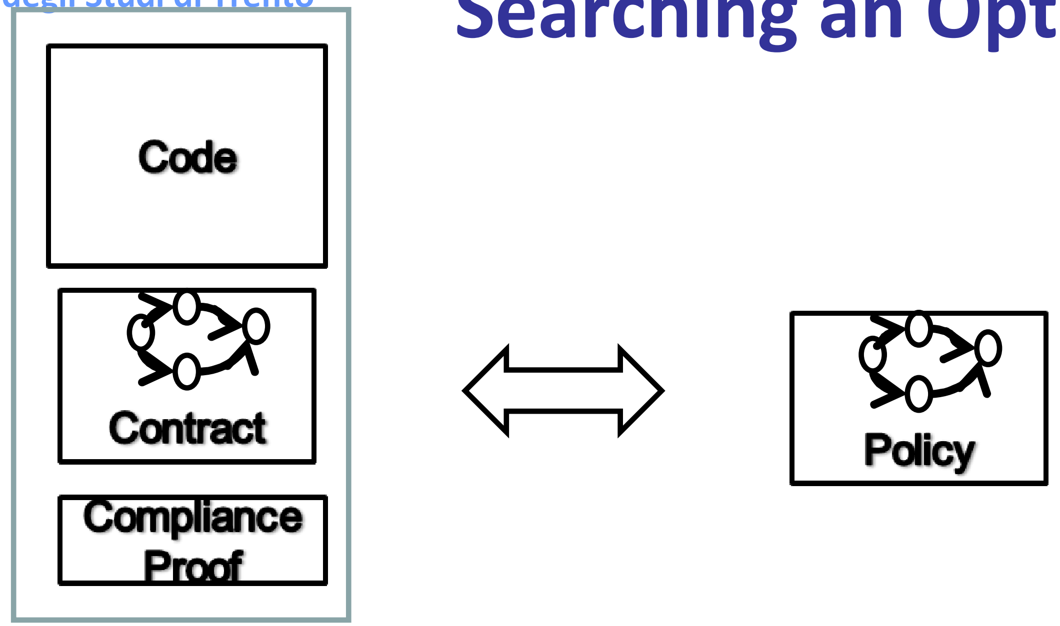
Università degli Studi di Trento

Optimizing Security Policy or Rewriter



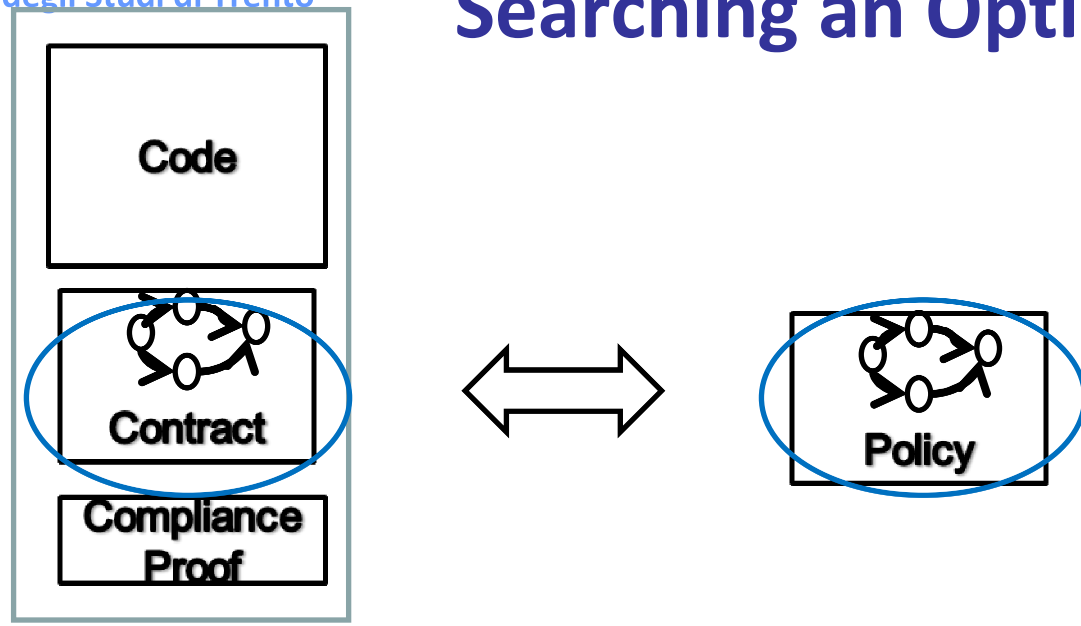
- **Security Automata SFI Implementation (SASI) [Erlingson-etal-NSPW'99]**
  - **Minimizing TCB by working at the level of object code**
- **Trade off between moving more processes out of trusted part and the complexity of the whole process [Hamlen-Thesis'06]**
- **Efficient IRM Enforcement [Yan-etal-ASIACCS'09]**
  - **a constrained representation of history-based access control policies**
  - **exploit the structure of this policy representation**
  - **extended into a distributed optimization protocol**

Searching an Optimized Policy



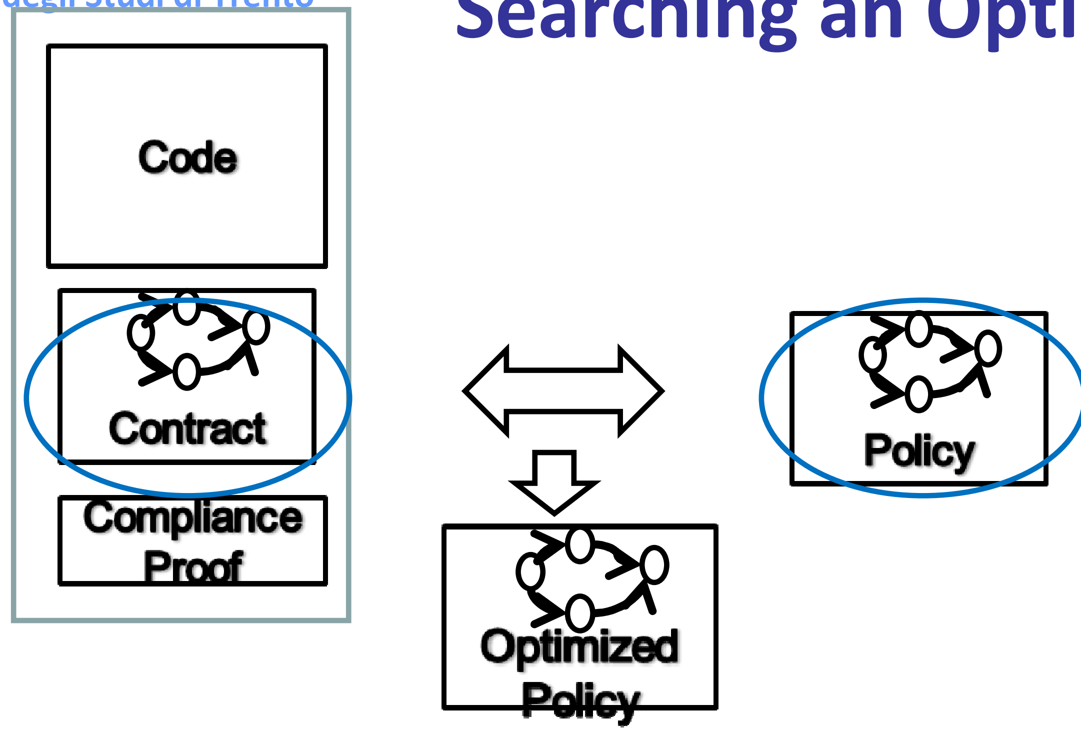
- **Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM O derived from P with respect to C when:**
  - every security-relevant event invoked by the intersection of O and C can also be invoked by P [sound]
  - O has smaller or equal number of transitions or states compared to P [optimal]

Searching an Optimized Policy



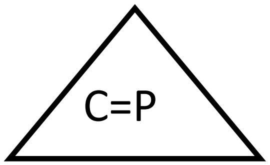
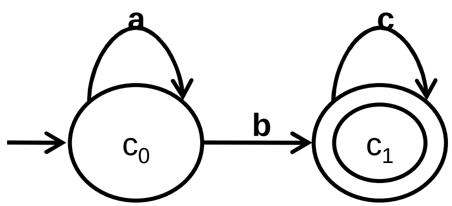
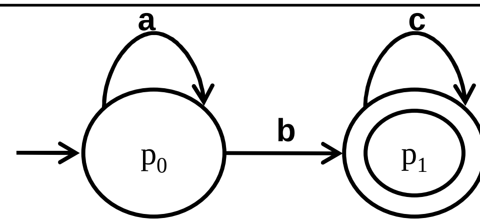
- **Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM O derived from P with respect to C when:**
  - every security-relevant event invoked by the intersection of O and C can also be invoked by P [sound]
  - O has smaller or equal number of transitions or states compared to P [optimal]

Searching an Optimized Policy

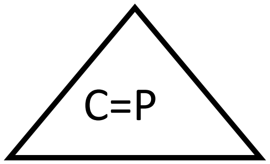
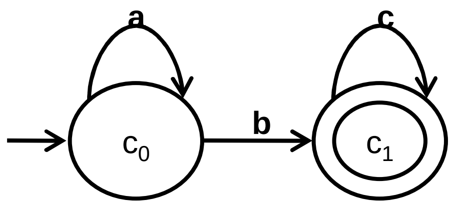
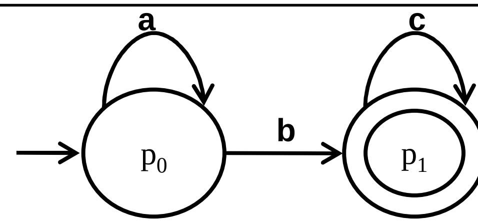
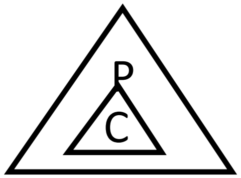
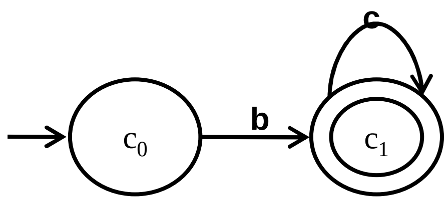
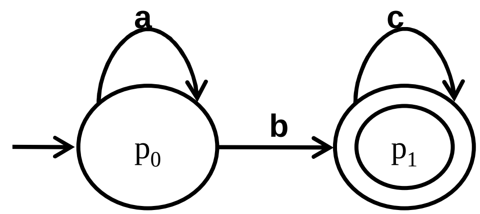


- **Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM O derived from P with respect to C when:**
  - every security-relevant event invoked by the intersection of O and C can also be invoked by P [sound]
  - O has smaller or equal number of transitions or states compared to P [optimal]

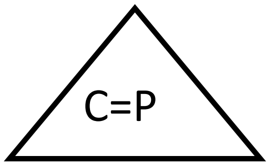
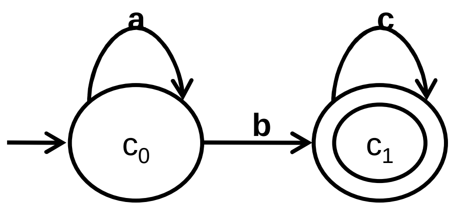
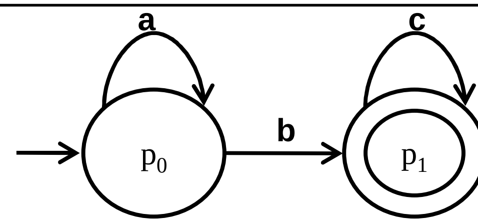
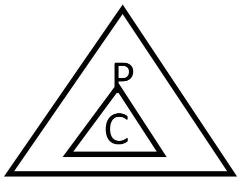
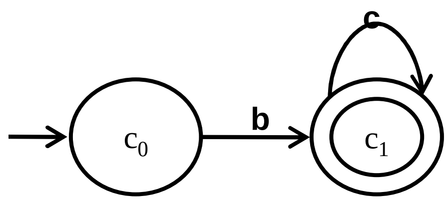
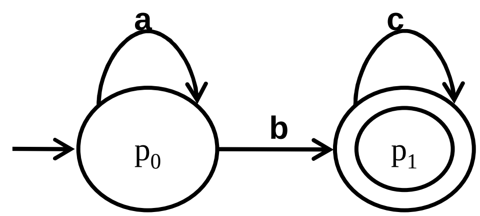
Inline Type Examples

Inline-type	Contract	Policy
		

Inline Type Examples

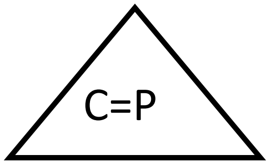
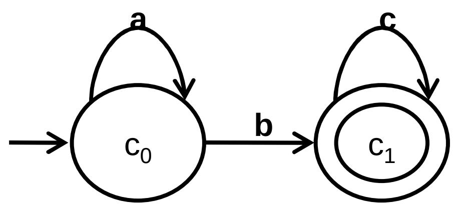
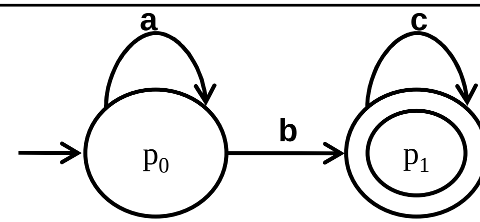
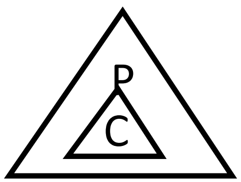
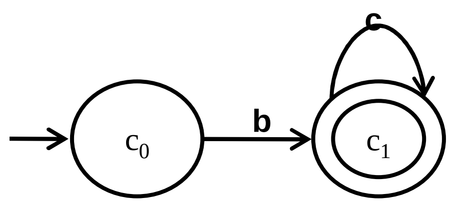
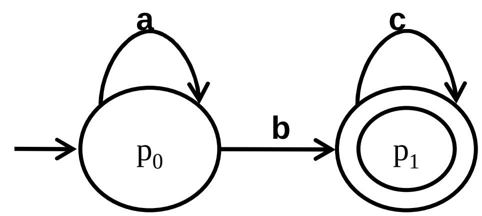
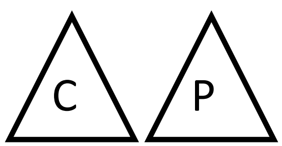
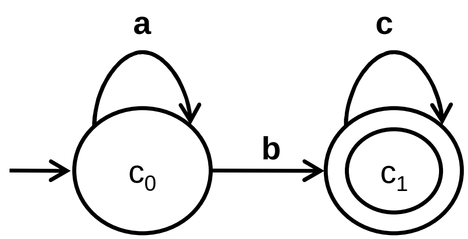
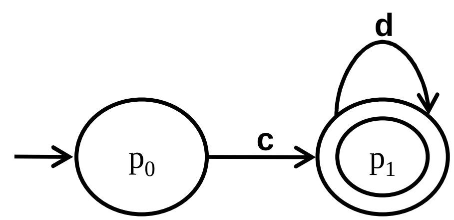
Inline-type	Contract	Policy
		
		

Inline Type Examples

Inline-type	Contract	Policy
		
		

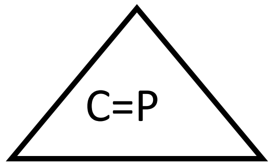
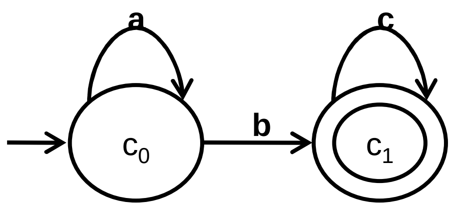
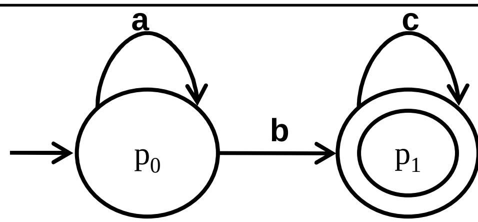
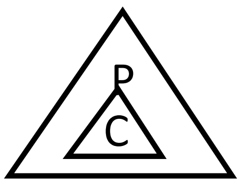
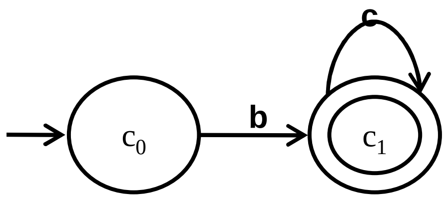
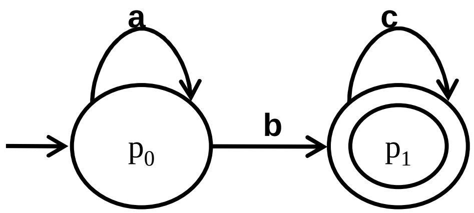
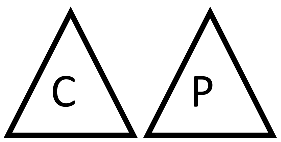
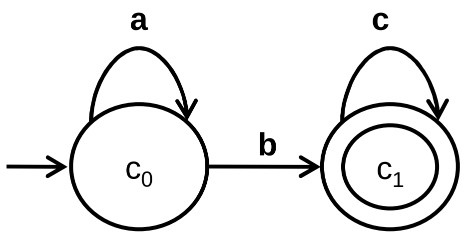
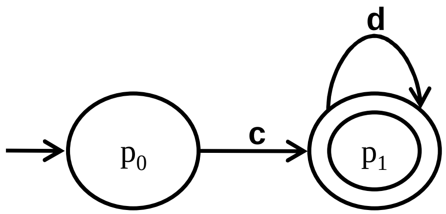
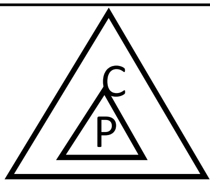
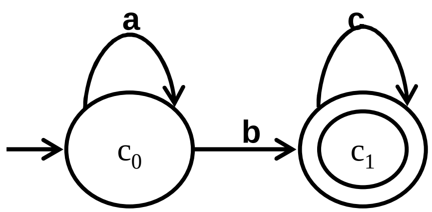
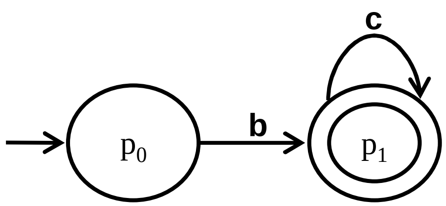
Inline nothing

Inline Type Examples

Inline-type	Contract	Policy
		
		
		

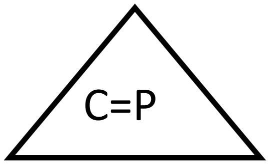
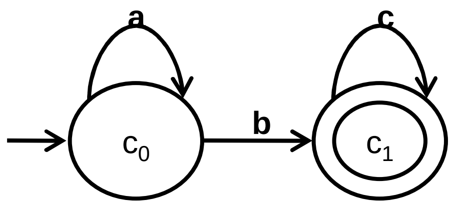
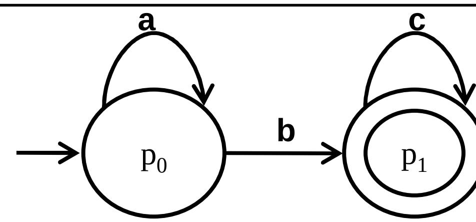
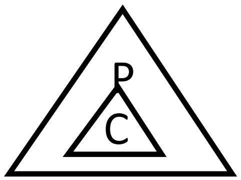
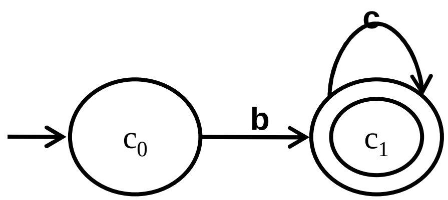
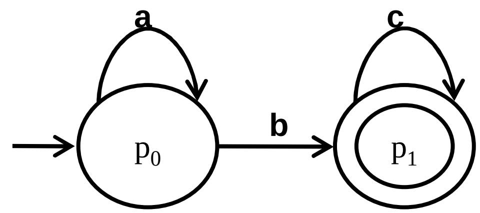
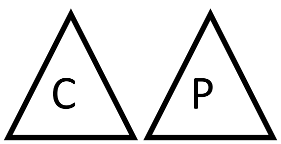
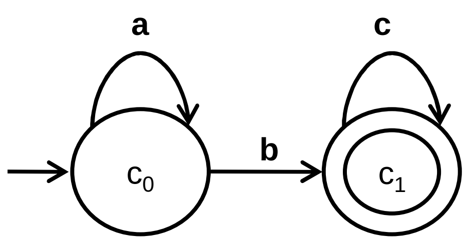
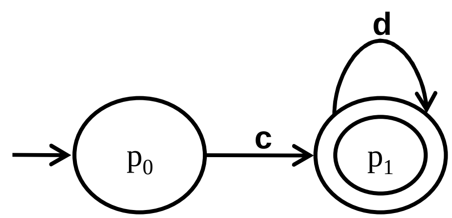
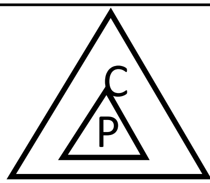
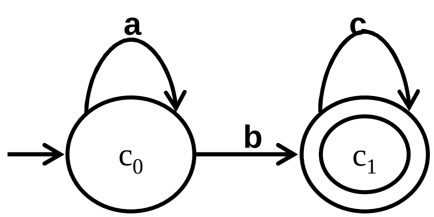
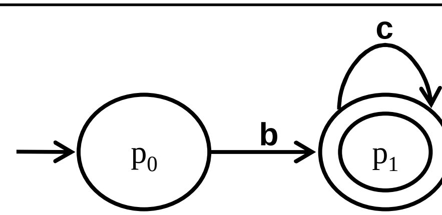
Inline nothing

Inline Type Examples

Inline-type	Contract	Policy
		
		
		
		

Inline nothing

Inline Type Examples

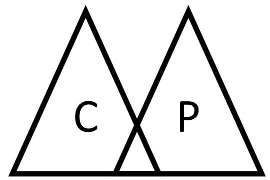
Inline-type	Contract	Policy
		
		
		
		

Inline nothing

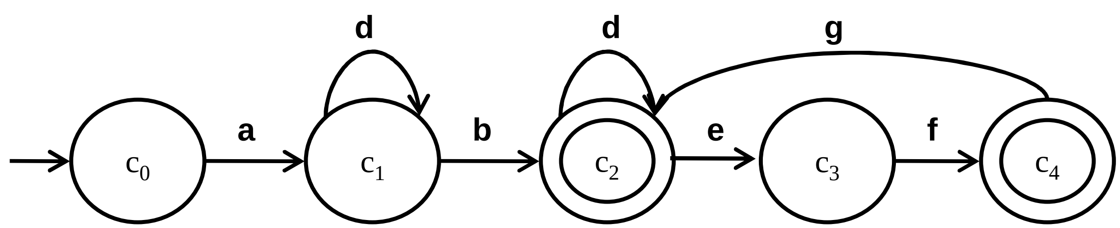
Inline all

Optimization Example

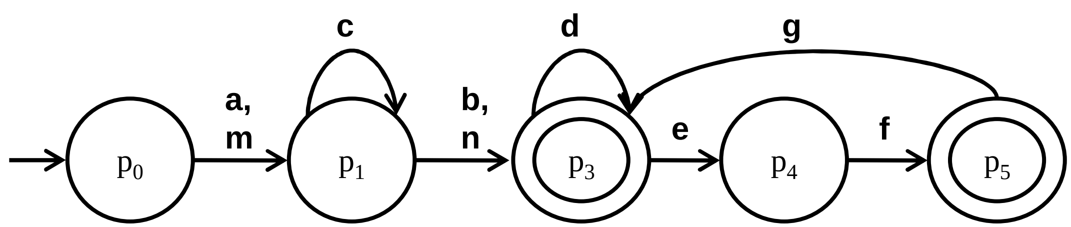
Inline-type



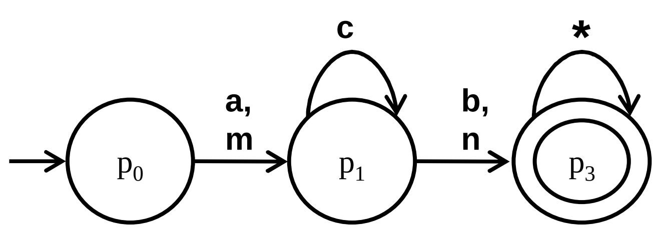
Contract



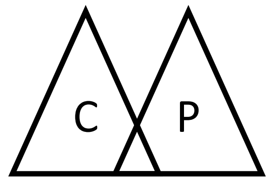
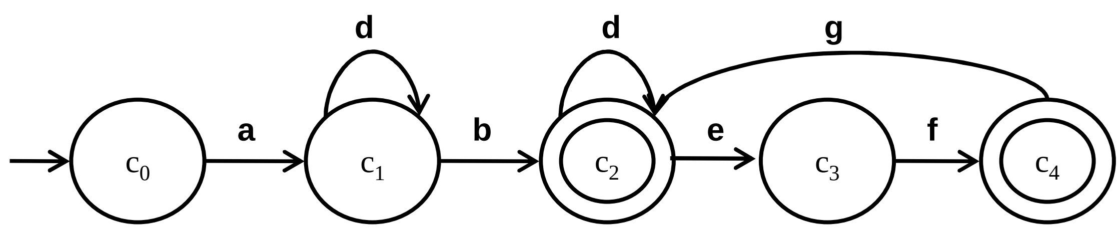
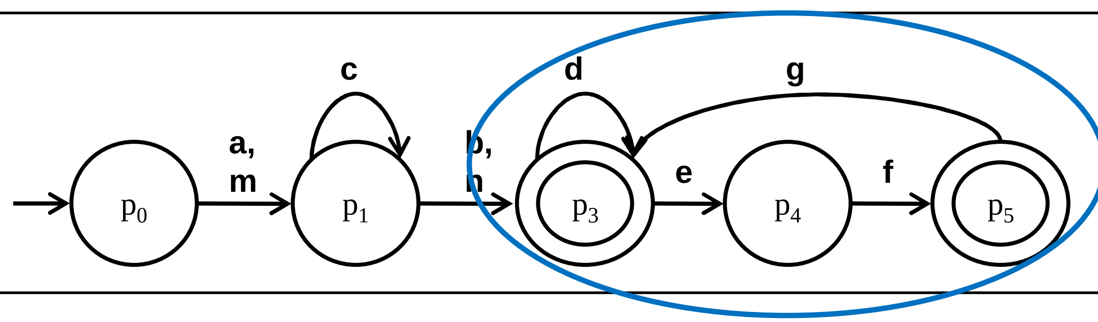
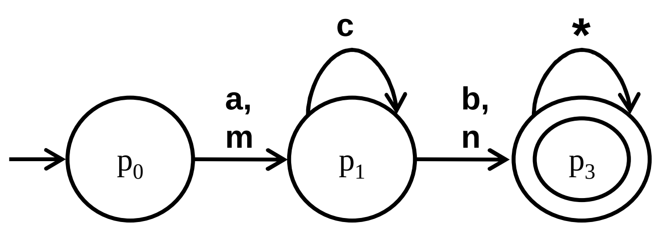
Policy



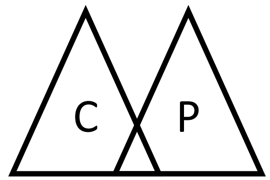
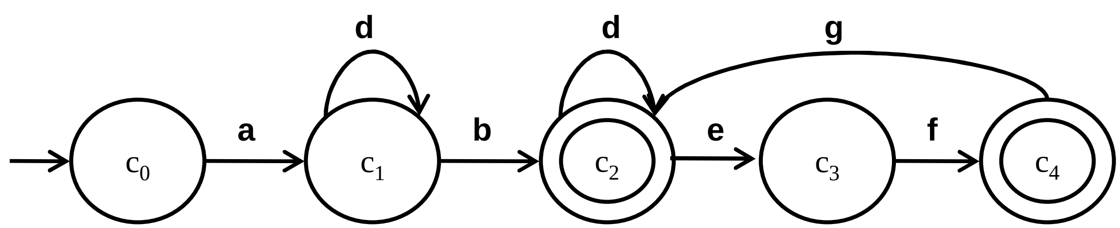
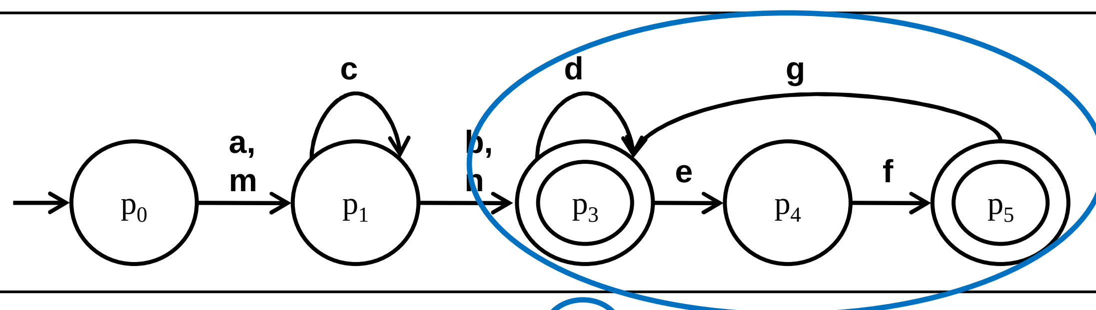
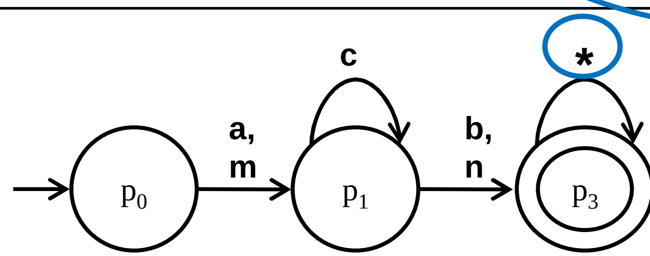
Optimized Policy



Optimization Example

Inline-type	
Contract	
Policy	
Optimized Policy	

Optimization Example

Inline-type	
Contract	
Policy	
Optimized Policy	

Publications

Journals:

- **[DJM+08] L. Desmet, W. Joosen, F. Massacci, P. Philippaerts, F. Piessens, I. Siahaan and D. Vanoverberghe. Security-by-contract on the .NET platform. In *Information Security Technical Report, Volume 13 Issue 1*, 2008.**
- **[BDM+09] N. Bielova, N. Dragoni, F. Massacci, K. Naliuka and I. Siahaan. Matching in Security-by-Contract for Mobile Code. In *Journal of Logic and Algebraic Programming***

Conferences:

- **[BTDS08] N. Bielova, M. DallaTore, N. Dragoni, and I. Siahaan. Matching Policies with Security Claims of Mobile Applications. In *Proc. of The 3rd International Conference on Availability, Reliability and Security (ARES'08)***

Publications

Workshops:

- [DMNS07] N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan. Security-by-Contract: Toward a Semantics for Digital Signatures on Mobile Code. In Proc. of The 4th European PKI Workshop (EuroPKI'07)
- [MS07] F. Massacci and I. Siahaan. Matching midlet's security claims with a platform security policy using automata modulo theory. In Proc. of The 12th Nordic Workshop on Secure IT Systems (NordSec'07)
- [MS08] F. Massacci and I. Siahaan. Simulating Midlet's Security Claims with Automata Modulo Theory. In Proc. of ACM SIGPLAN 3rd Workshop on Programming Languages and Analysis for Security (PLAS 2008)
- [BMS08a] N. Bielova and I. Siahaan. Testing Decision Procedures for Security-by-Contract. In Proc. of Joint Workshop on Foundations of Computer Security, Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (FCS-ARSPA-WITS'08)
- [MS09] F. Massacci and I. Siahaan. Optimizing IRM with Automata Modulo Theory. In 5th International Workshop on Security and Trust Management (STM 2009).

Conclusions

- **Security policies of both safety and liveness properties**
- **Mechanism for defining a general security policies (not platform-specific)**
- **Mechanism for representing an infinite structure as a finite structure**
- **Goal:**
  - to provide contract-policy matching
  - issues: small memory footprint, efficient computations
  - the tractability limit is the complexity of the satisfiability procedure for the background theories used to describe expressions
- **Results:**
  - Contract-policy matching problem for AMT using language inclusion and simulation
  - Policy optimization problem for AMT using fair simulation

Thank you

References

- J.R. Büchi, “On a decision method in restricted second-order arithmetic. ”, Int. Cong. on Logic, Methodology and Philosophy of Science, 1962.
- U. Erlingsson, F.B. Schneider, “SASI Enfocement of Security Policies: A Retrospective”, New Security Paradigm Workshop 1999
- U. Erlingsson, F. B. Schneider, “IRM Enforcement of Java Stack Inspection”, IEEE Symposium on Security and Privacy 2000
- K. Hamlen, “Security policy enforcement by automated program-rewriting,” Ph.D. thesis, Cornell University, 2006.
- F. Schneider, “Enforceable Security Policies”, ACM Transactions on Information and System Security, Vol. 3, No. 1, February 2000
- R. Sebastiani,“Lazy Satisability Modulo Theories”, Journal on Satisability, Boolean Modeling and Computation 3 (2007) 141-224
- F. Yan, P.W.L. Fong , “Efficient IRM Enforcement of History-Based Access Control Policies.”, ASIACCS 2009