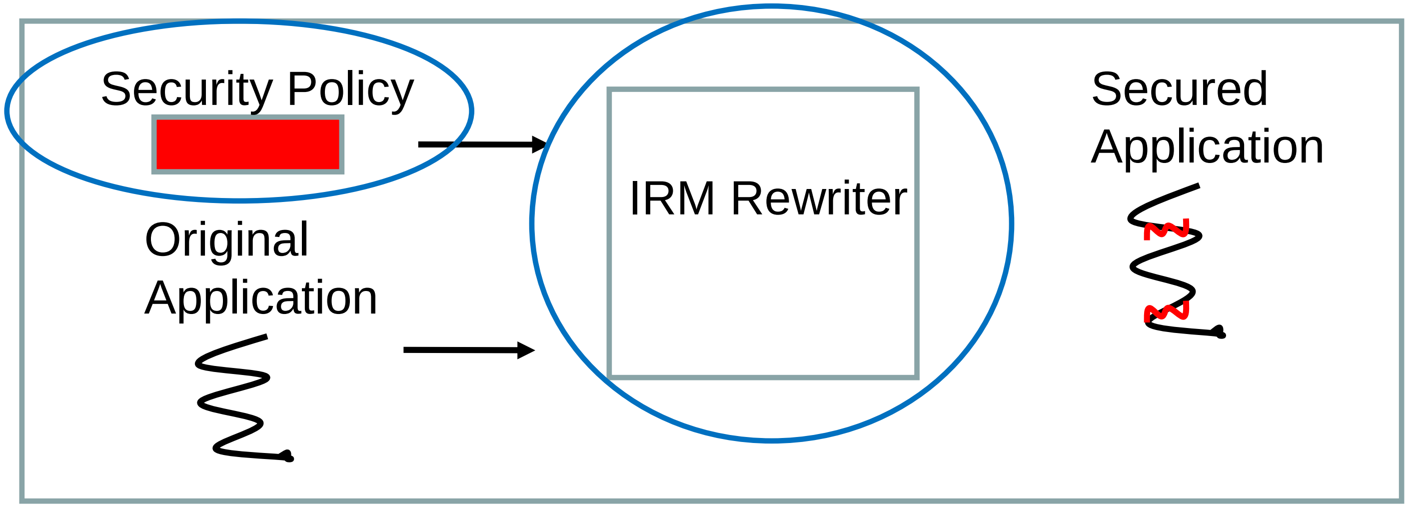
Università degli Studi
di Trento

Inline-Reference Monitor Optimization using Automata Modulo Theory (AMT)

Fabio Massacci

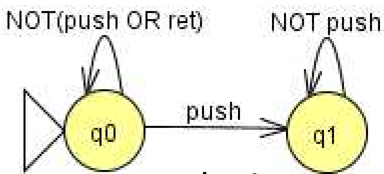
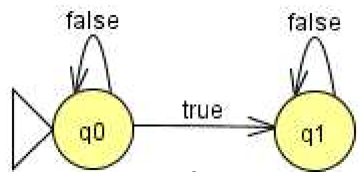
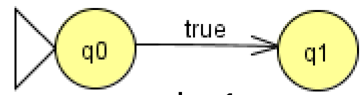
<u>Ida Siahaan</u>

Inlined Reference Monitors



- Policy Enforcement Toolkit (PoET)
  - **Implementing IRMs for Java Virtual Machine Language (JVML) applications**
  - **Primary concern: trusted computing base (TCB) 17,500 loc Java source code**
  - **U. Erlingsson, F. B. Schneider, "IRM Enforcement of Java Stack Inspection", IEEE Symposium on Security and Privacy 2000**

Optimizing Security Policy or Rewriter

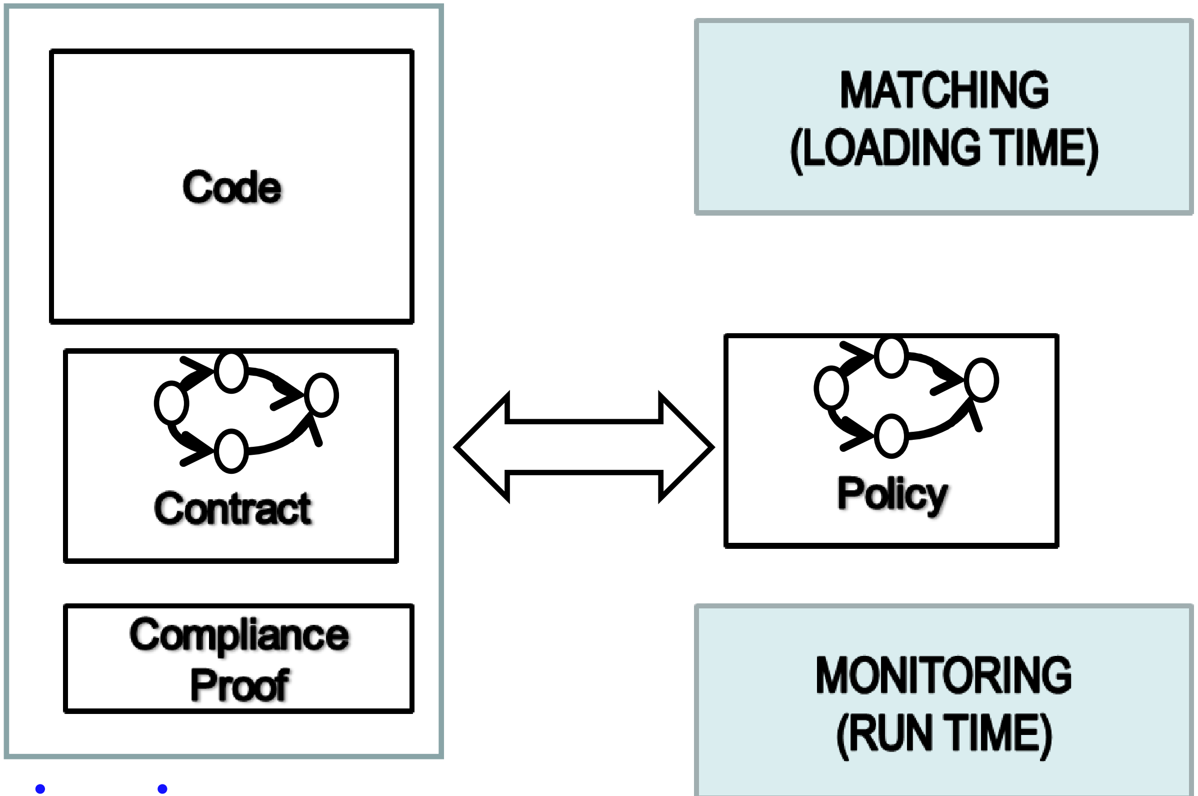
Insert Security Automata	Evaluate transitions	Simplify Automata	Compile Automata
 push r1	 push r1	 push r1	**if state==q0** **then state:=q1** **else ABORT** push r1

- Security Automata SFI Implementation (SASI)
  - **Implementing IRMs for x86 and JVML**
  - **Minimizing TCB by working at the level of object code**
  - **Ulfar Erlingsson, Fred B. Schneider, "SASI Enfocement of Security Policies: A Retrospective", New Security Paradigm Workshop 1999**

Optimizing Security Policy or Rewriter

- **Trade off between moving more processes out of trusted part and the complexity of the whole process**
  - **K. Hamlen, "Security policy enforcement by automated program-rewriting," Ph.D. thesis, Cornell University, 2006.**
- **Efficient IRM Enforcement**
  - **a constrained representation of history-based access control policies**
  - **exploit the structure of this policy representation**
  - **extended into a distributed optimization protocol**
  - **F. Yan, P.W.L. Fong , "Efficient IRM Enforcement of History-Based Access Control Policies.", ASIACCS 2009**

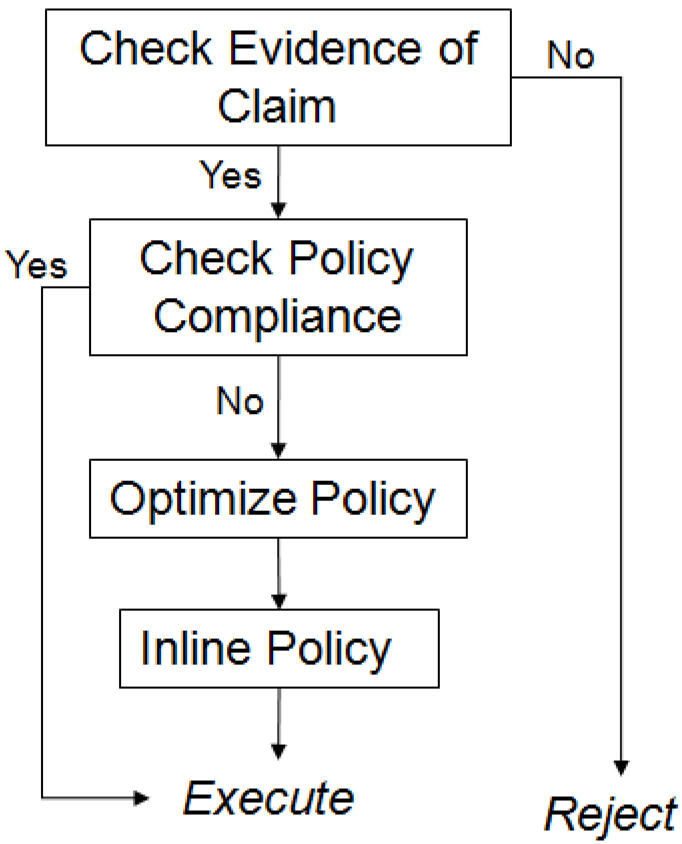
Security by Contract (SxC)



- SxC device view
  - N. Bielova, N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan, "Matching in security-by-contract for mobile code", J. of Logic and Algebraic Programming 2009

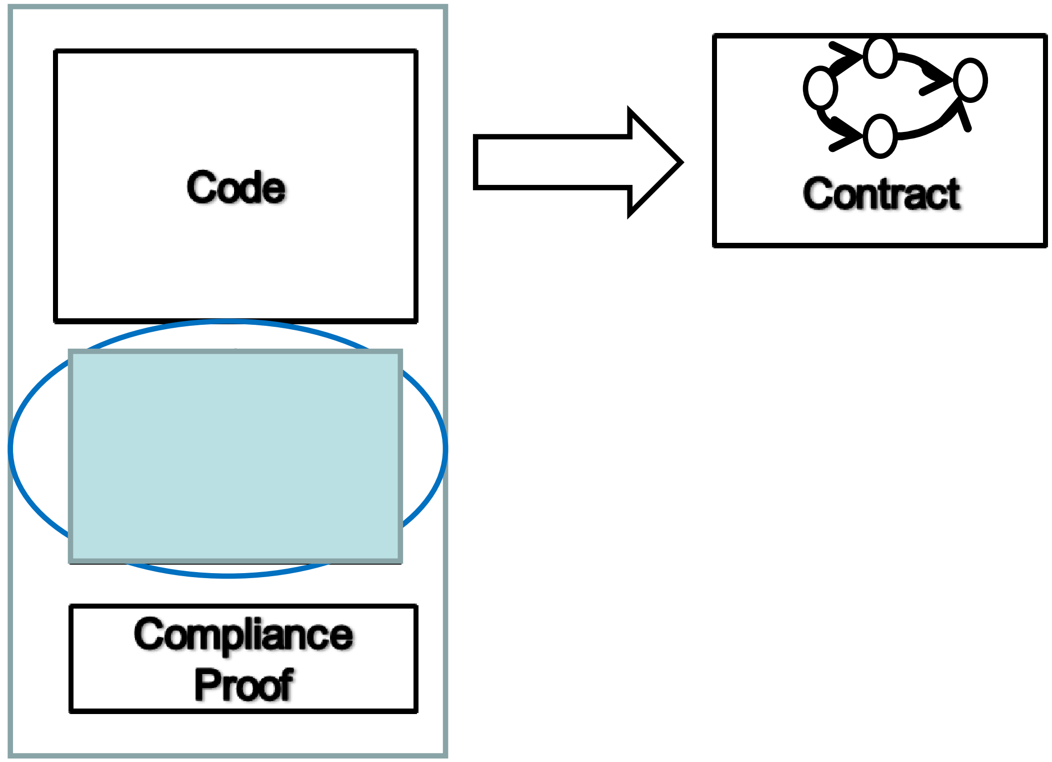
Università degli Studi
di Trento

IRM Optimization



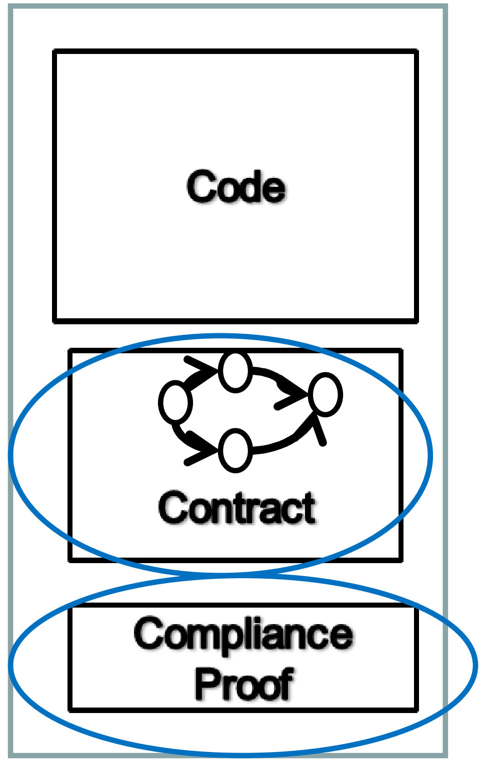
Given an (un)trusted code and a policy that a platform specifies to be inlined, how can we obtain an optimized IRM ?

Components of IRM Optimization



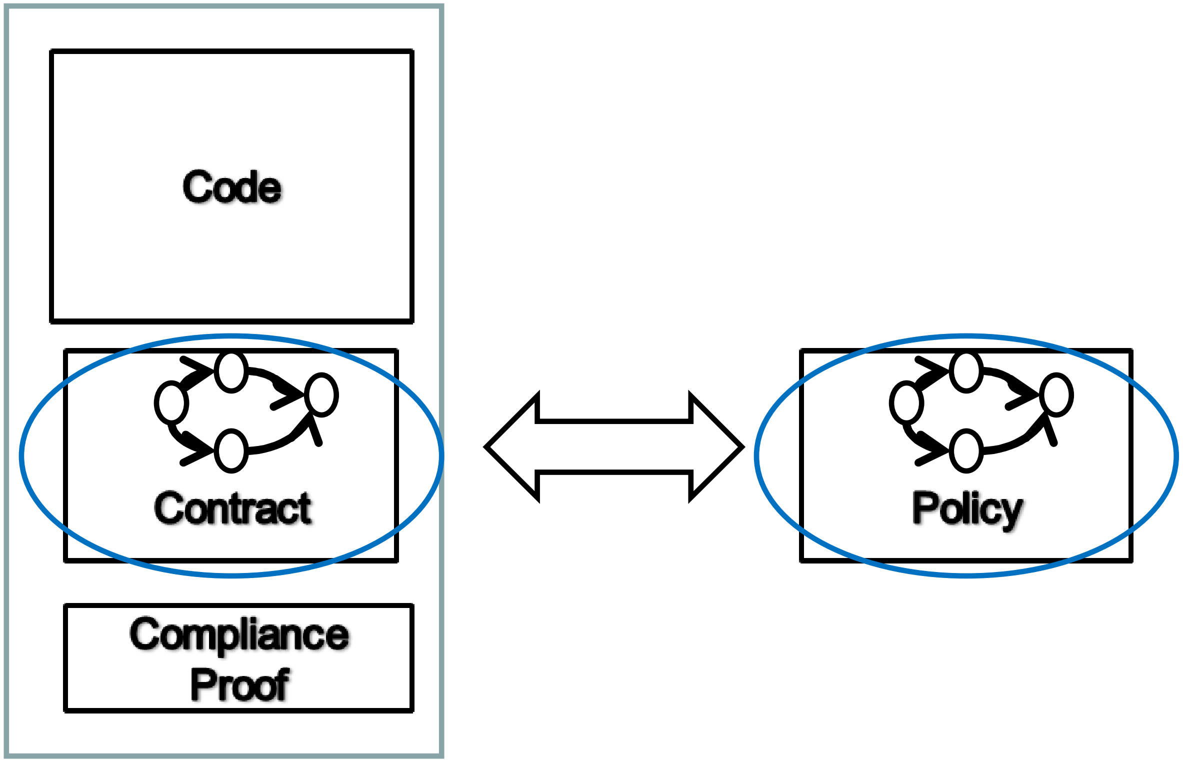
- **Contract Extractor**
  - **extract *security relevant behaviors* from code**

Components of IRM Optimization



- **Claim Checker**
  - ***verify that the claimed contract complies to the code***
  - ***digitally signed by a trusted code provider***

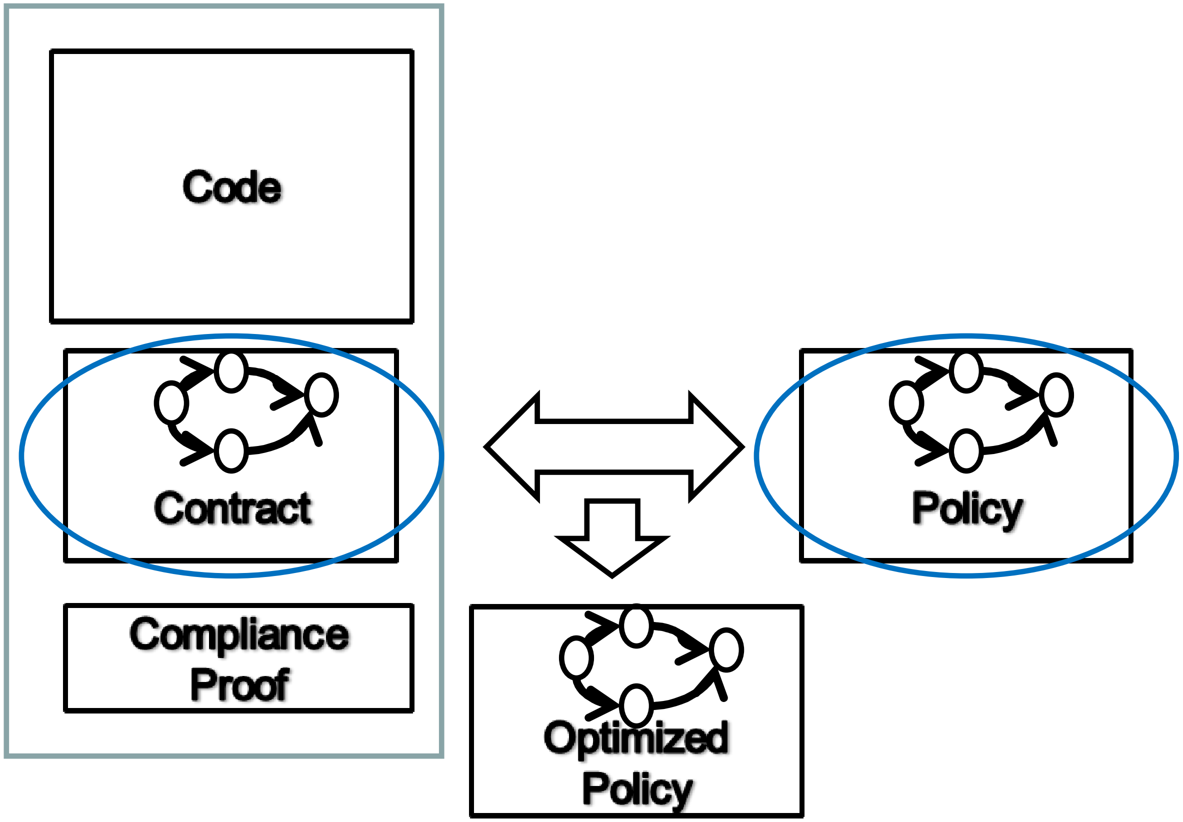
Components of IRM Optimization



- **Simulation Checker**
  - *check a policy simulates a contract*

Università degli Studi
di Trento

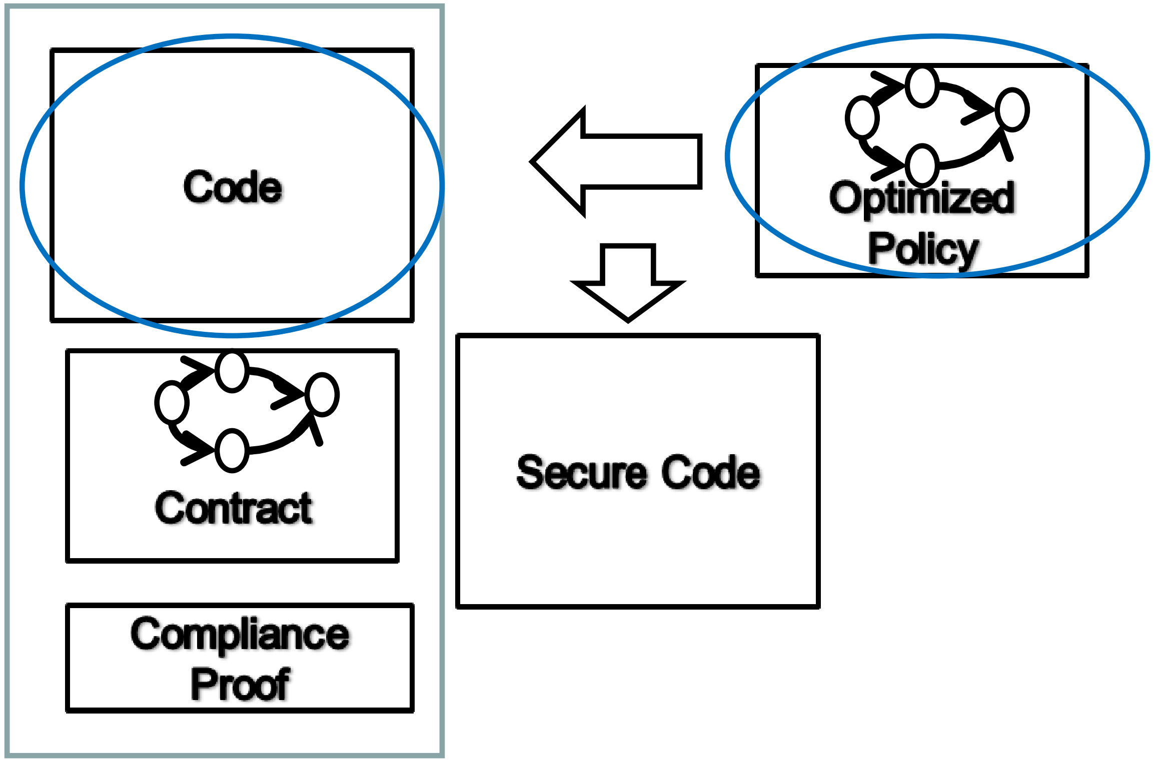
Components of IRM Optimization



- **Optimizer**
  - ***discharge behaviors which are already enforced by code***

Università degli Studi
di Trento

Components of IRM Optimization

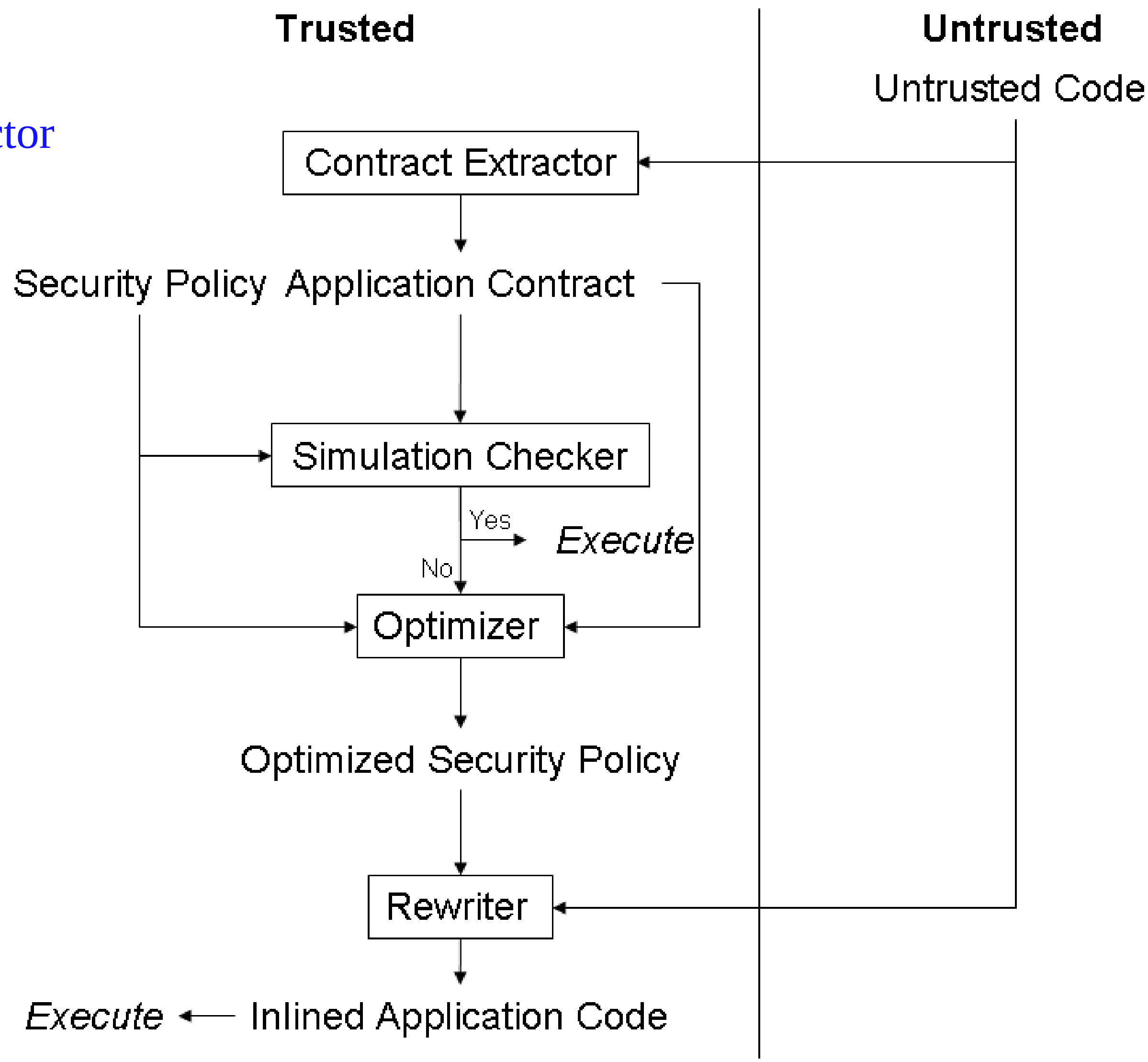


- **Rewriter**
  - ***inject policy to the code***

IRM Optimization Models

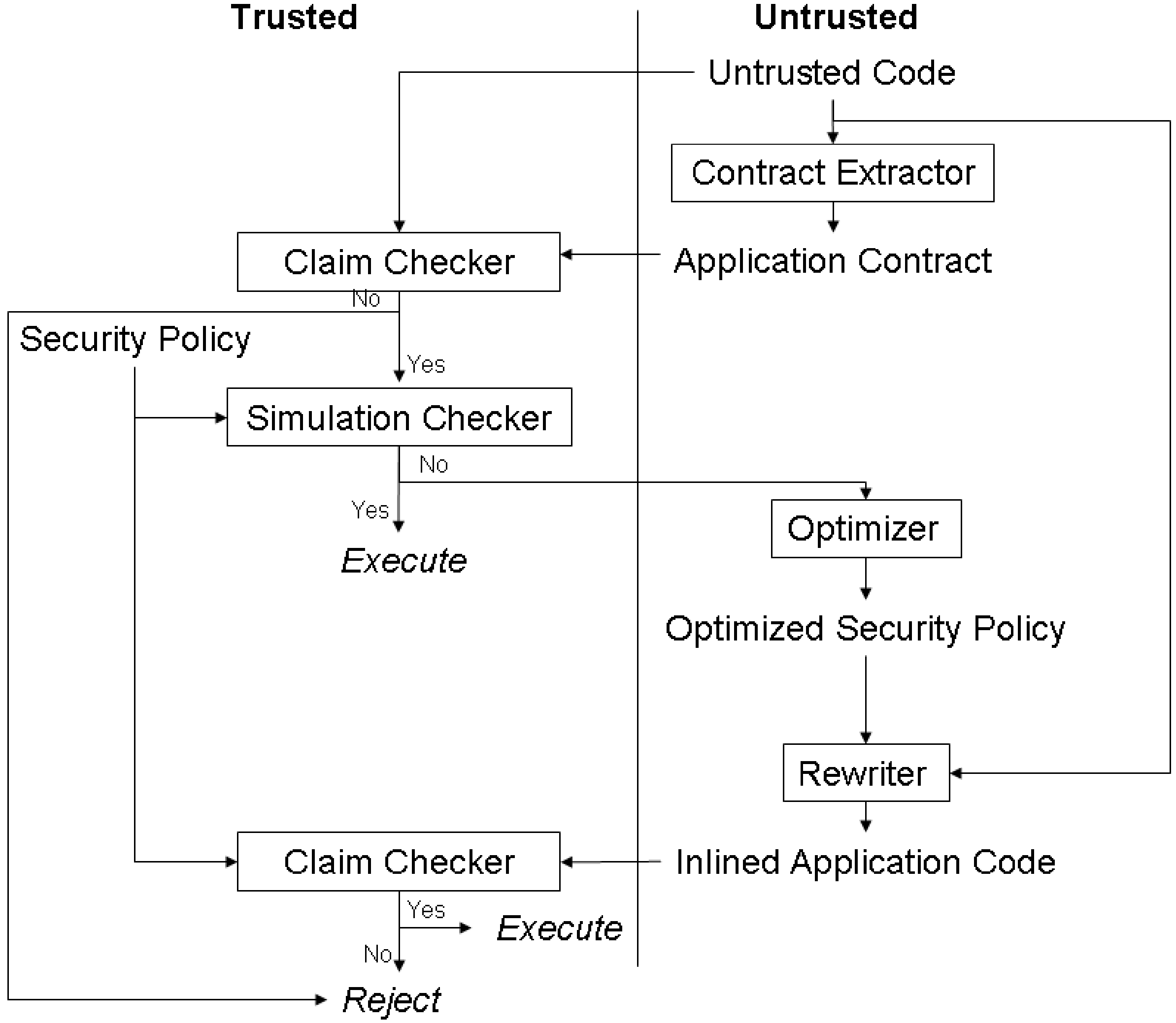
Rewriter on Trusted part

Model1:
Contract Extractor
on Trusted part



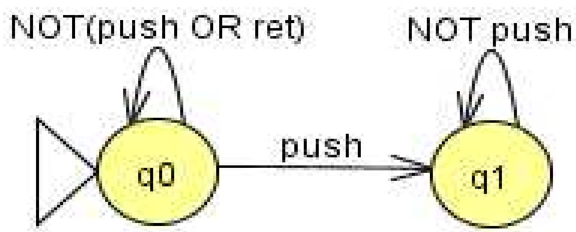
Optimizer and Rewriter on Untrusted part

Model6:
Contract Extractor
on Untrusted part



Automata Modulo Theory (AMT)

Security Automata



- **A class of Büchi automata that accept safety properties (recognizers)**
  - a countable set Q of *automaton states*,
  - a countable set $Q_0 \subseteq Q$ of *initial automaton states*,
  - a countable set I of *input symbols*, and
  - a *transition function* $\delta : (Q \times Q) \rightarrow 2^Q$
  - F. Schneider, "Enforceable Security Policies", ACM Transactions on Information and System Security, Vol. 3, No. 1, February 2000

Università degli Studi
di Trento

Edit Automata

- **Truncation automaton (recognizer)**
  - terminate application
- **Suppression automaton (transducer)**
  - truncation automaton + suppress undesired or dangerous actions without necessarily terminating the program
- **Insertion automaton (transducer)**
  - truncation automaton + insert additional actions into the event stream
- **Edit automata = Suppression automaton + Insertion automaton**
  - Jay Ligatti, Lujo Bauer, David Walker, "Enforcement Mechanisms for Run-time Security Policies?", Int J Inf Secur (2005) 4

Automata Modulo Theory (AMT)

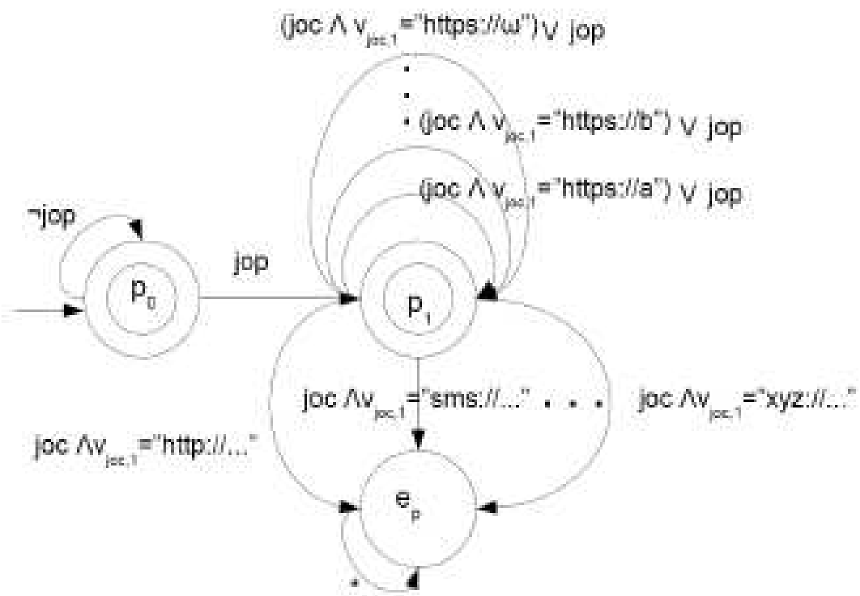
- **AMT = Büchi automata + Satisability Modulo Theories (SMT)**
  - a set E of formulae in the language of the theory T as *input symbols*
  - a finite set Q of *automaton states*,
  - an *initial state* $q_0 \in Q$,
  - a set $F \subseteq Q$ of *accepting states*, and
  - a *labeled transition function* $\delta : (Q \times E) \rightarrow 2^Q$
  - F. Massacci, I. Siahaan, "Matching midlet's security claims with a platform security policy using automata modulo theory.", NordSec'07

Università degli Studi di Trento

Satisability Modulo Theories (SMT)

- The problem of deciding the satisability 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))**
- Examples of SMT tools:
  - **Z3**
  - **MathSAT**
- Primary interest for SMT(T) when T is a combination of two or more theories $T_1,\ldots,T_n$.
  - **Example of an atom: f(x + 4y) = g(2x - y)**
  - **R.Sebastiani,“Lazy Satisability Modulo Theories”, Journal on Satisability, Boolean Modeling and Computation 3 (2007) 141-224**

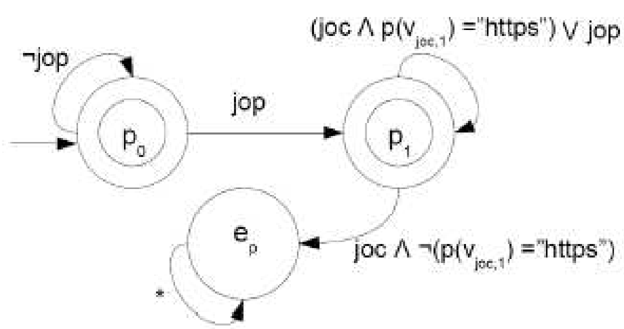
Example of AMT



(a) Infinite Transitions Security Policies

$$
\begin{array}{rcl}
joc(v_{joc,1}) & \doteq & \texttt{io.Connector.open}(url) \\
jop & \doteq & \texttt{pim.PIM.openPIMList}(\ldots) \\
q & \doteq & \texttt{io.Connector.}type \\
 & & \text{is protocol type e.g. "http"} \\
pr(q) = type & \doteq & \text{permission } q \text{is for protocol } type \\
p(url) = type & \doteq & url.\texttt{startsWith}(type)
\end{array}
$$

(b) Abbreviations for Java APIs



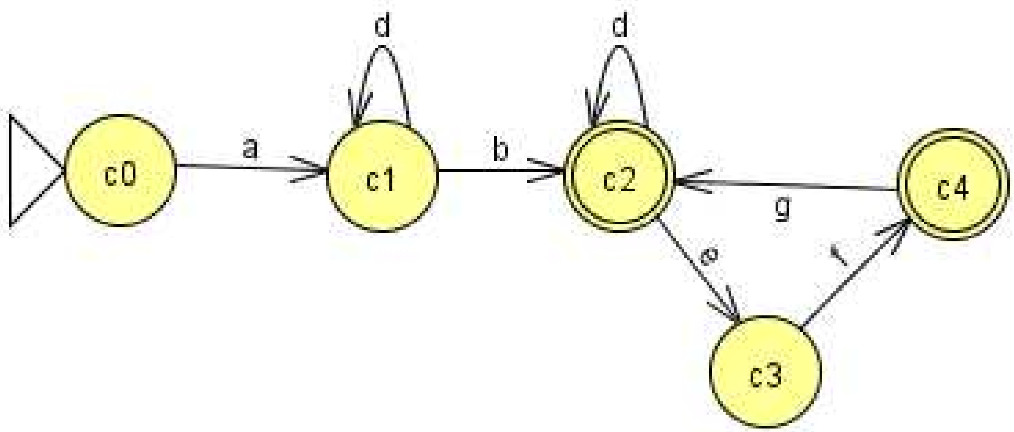
IRM Optimization using AMT

Searching an Optimized Policy

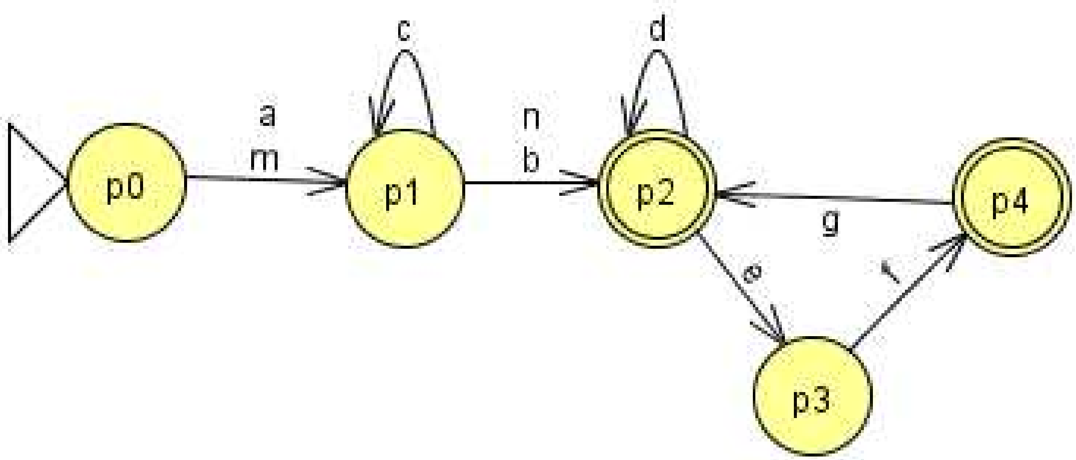
- **Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM OptP derived from P with respect to C when:**
  - every APIs invoked by the intersection of OptP and C can also be invoked by P [sound]
  - OptP is smaller than P with respect to C [optimal]

Università degli Studi
di Trento

Contract-Policy Example



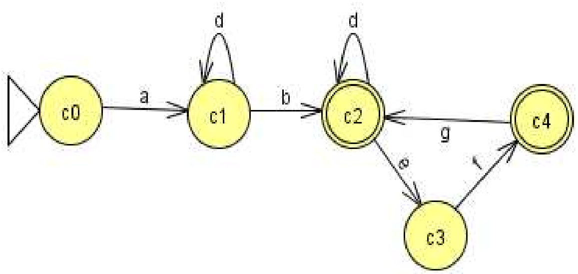
Contract



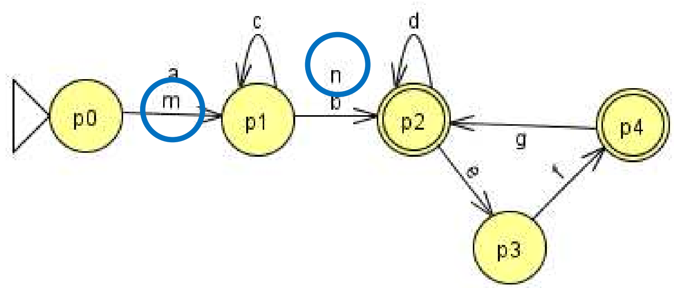
Policy

Università degli Studi
di Trento

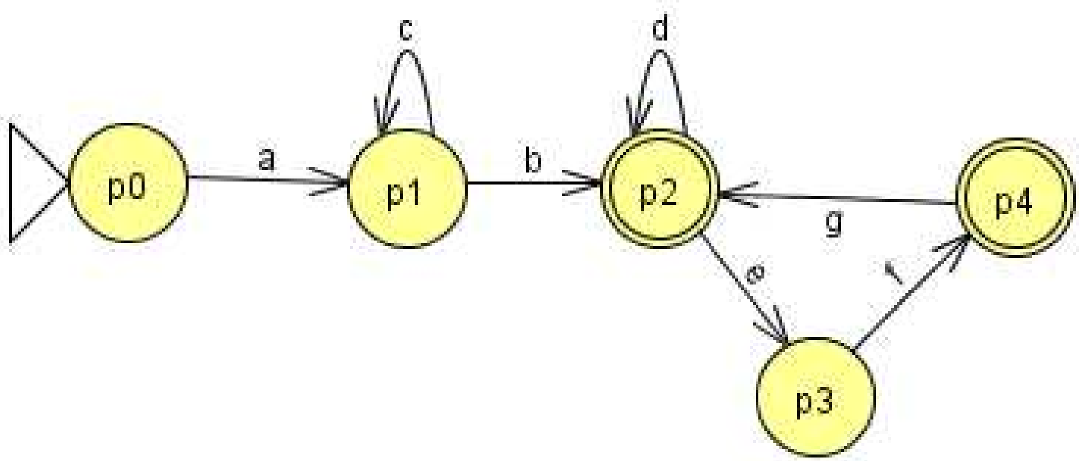
Removes non existing actions



Contract

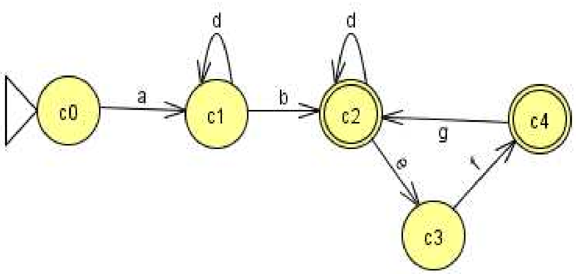


Policy

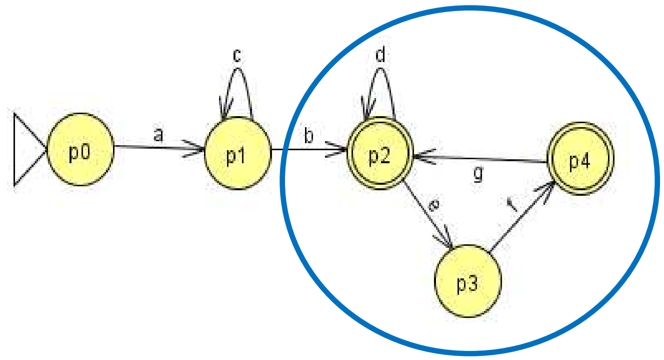


Optimize 1
Policy

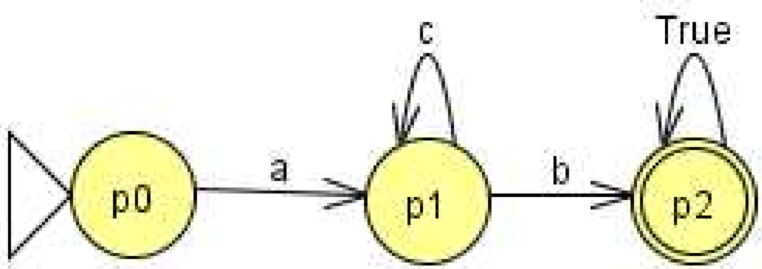
Removes already promised actions



Contract



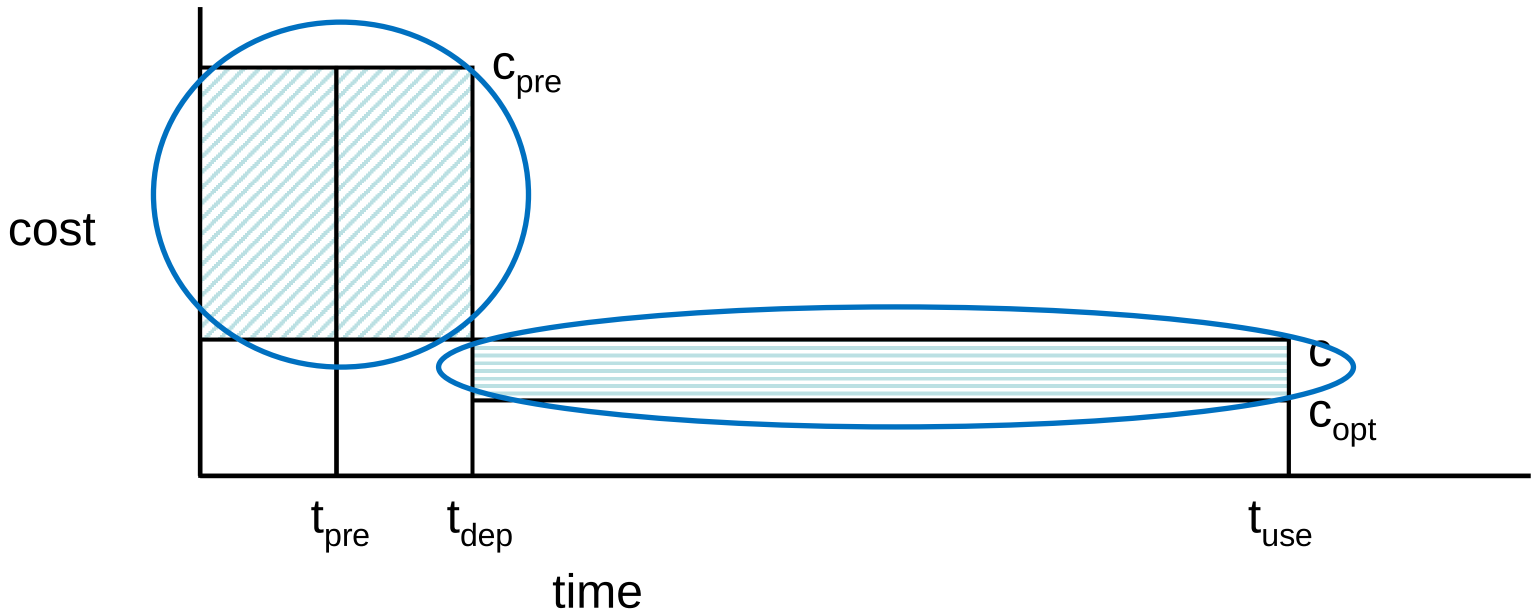
Optimize 1
Policy



Optimize 2
Policy

Future Work

- Implementation and study of IRM with or without optimization

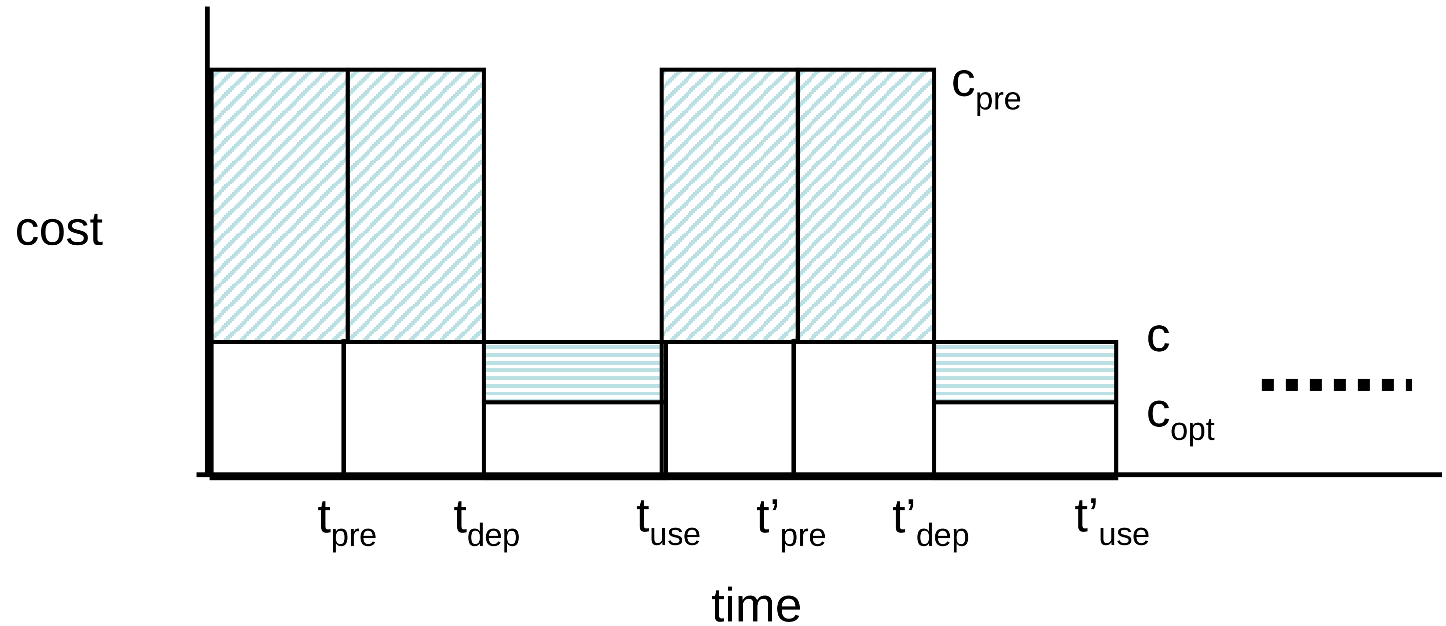


$c.t_{use} >> c_{pre}.(t_{pre} + t_{dep}) + c_{opt}.(t_{use} – (t_{pre} + t_{dep}))$

Assumption:
$t_{use} >> t_{pre}$
$t_{us} >> t_{dep}$

Future Work

- Effect of changes both in frequency (how often a code modified) and size (how much a code modified).



$$c.t_{use} \;??\; c_{pre}.(t_{pre} + t_{dep}) + c_{opt}.(t_{use} - (t_{pre} + t_{dep}))$$

Thank you