TECHNISCHE UNIVERSITÄT DARMSTADT

Expertise knowledge-based Policy Refinement Process

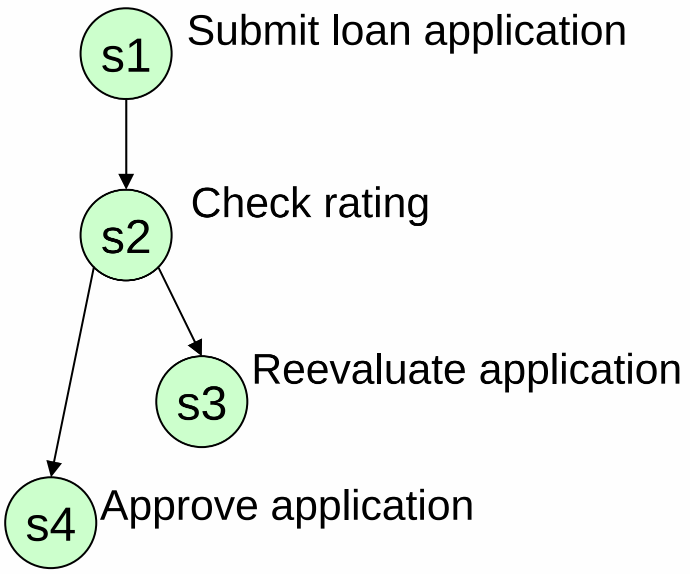
T. Rochaeli and C. Eckert

Technische Universität Darmstadt

Workflow and Kripke Model

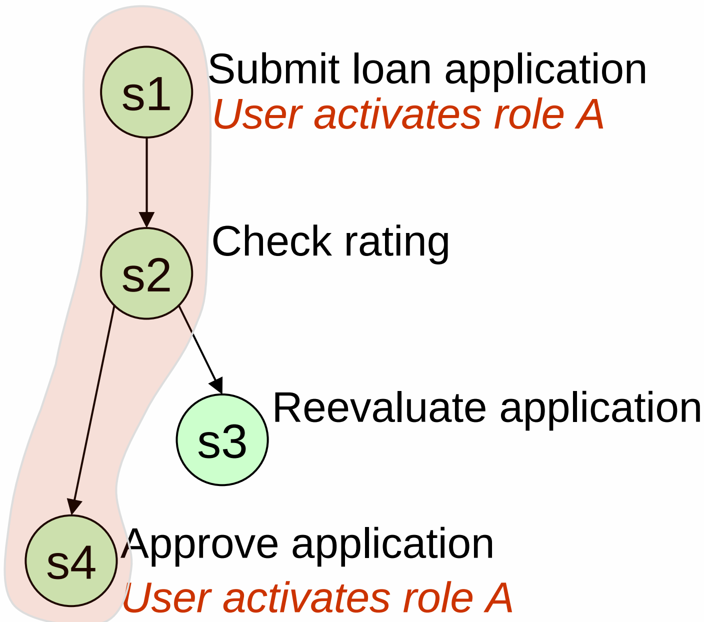
- Workflow: computerized facilitation or automation of a business process
- Kripke model represents the behavior of workflow
  - connected directed graph: nodes are states and edges are state transitions
  - state: snapshot of the workflow behavior
  - State transitions: possible next subsequent states
  - state labels: occurrence of events (i.e. task execution)

  - Formally, *M*: (W,R,L)
  - W, set of states
  - R $\subset$ W x W, set of state transitions
  - L: W $\rightarrow$ 2^{AP}, labeling function



The Gap between Design and Implementation

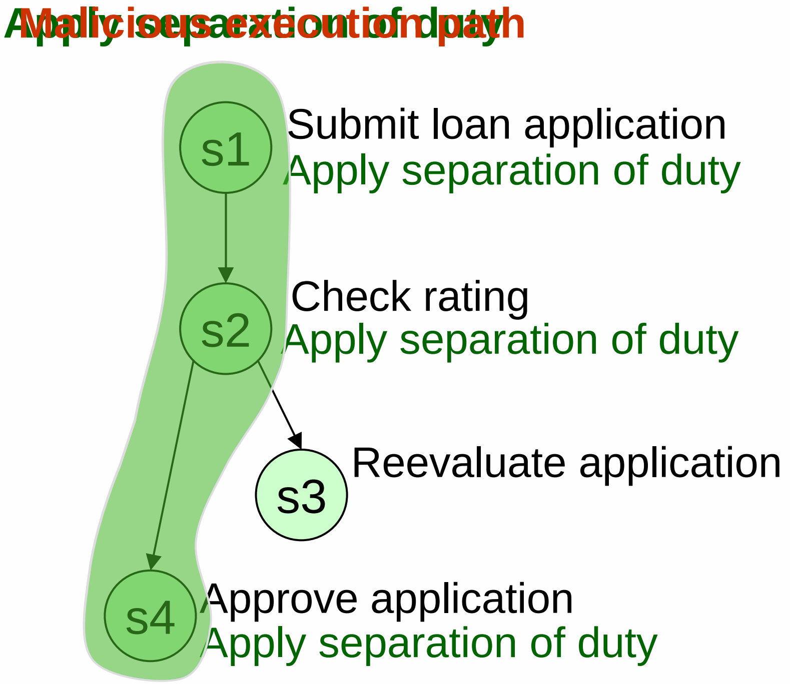
- Caused by the simplification of workflow design
  - Developer assumption: “Everything is just fine…”
- Model of workflow design
  - Consider only task’s execution
- Model of workflow implementation
  - any other events could also happen (i.e. role activation, user authentication)
- An example of malicious execution path (or trace)
  - Same role activates two sensitive tasks



Workflow Design
Implementation

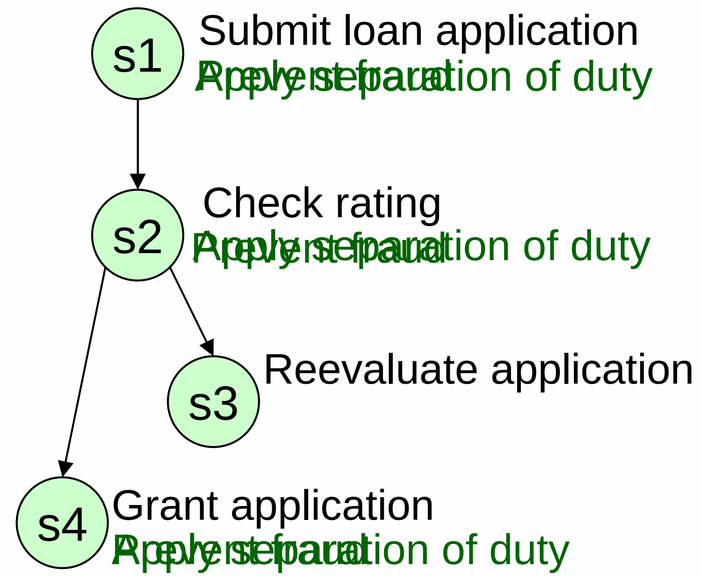
State Labels to Represent Security Policy

- Avoid the malicious execution path by specifying security policy in the shaded zone:
  - The security mechanism (separation of duty) should be applied within this execution path
- The shaded zone is represented by additional states label
- States labels along a fragment of execution path represent the security policy



Refining the State Labels

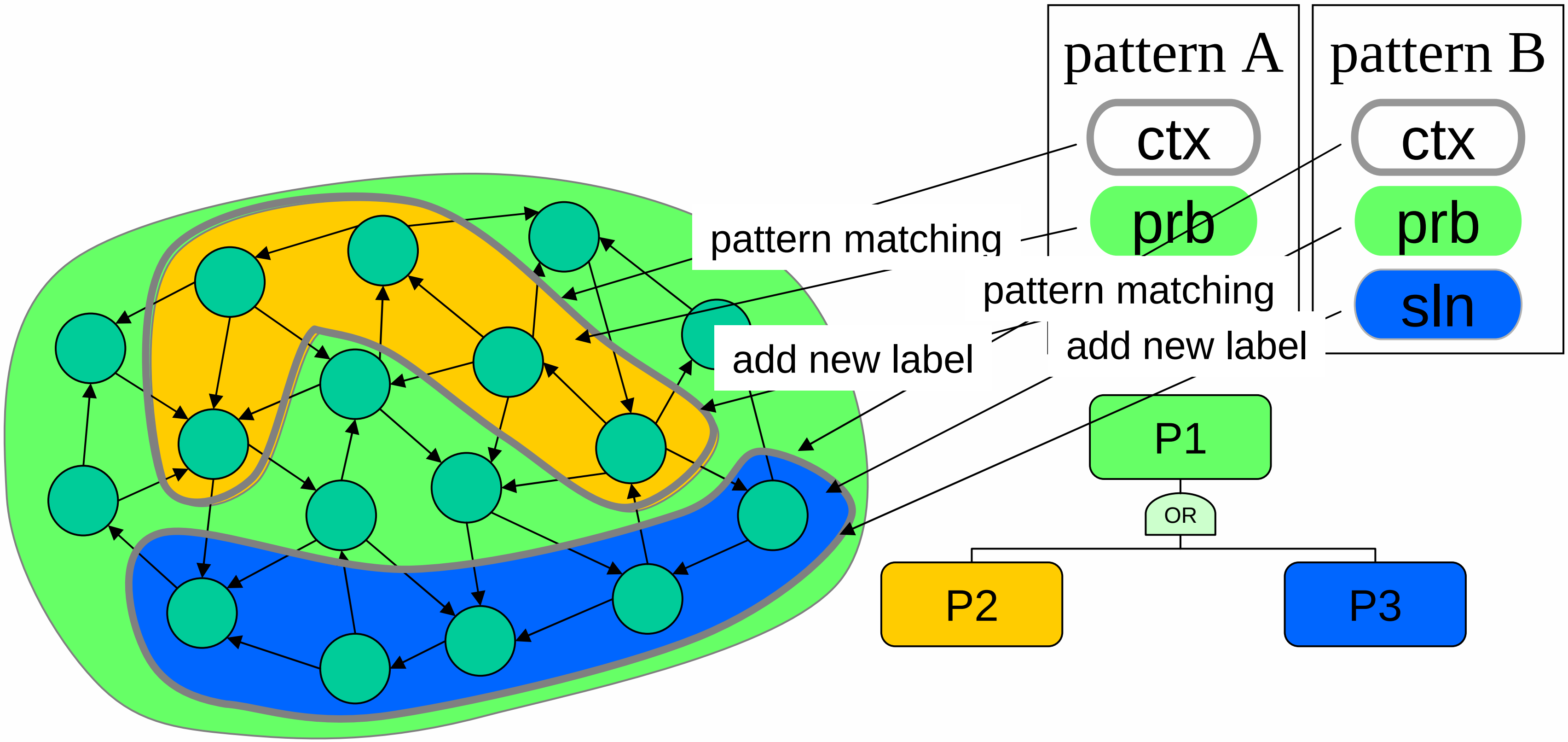
- Source of the policy refinement process: abstract policies
  - Originated from stakeholders' protection intent
  - Abstract state labels
- Target of the policy refinement process: concrete policies
  - Concrete state labels
  - Denote the execution path, in which the security mechanism should apply
- ➔ domain experts' knowledge is required!



Documenting the Experts' Knowledge

- Make use pattern paradigm
  - A pattern captures the best-practice *solution* to a *problem* in a certain *context*
- Three main parts of refinement pattern
  - Context: describes the execution path, in which the problem occurs
  - Problem: describes the abstract state labels
  - Solution: describes the less abstract state labels that should be defined within the context
- Formal representation (required for automated refinement process)
  - All parts of the pattern are represented by Linear-time Temporal Logic formulas
- Advantage:
  - Effective documentation and transfer of knowledge between domain experts
- Disadvantage:
  - The correctness of refined policies depends on the validity of the patterns

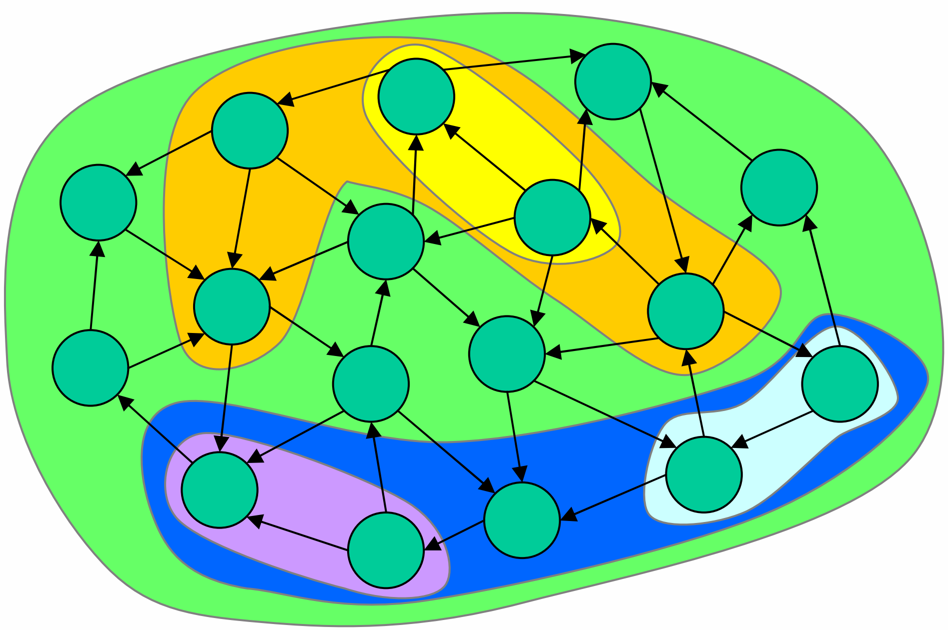
An Overview of Expertise Knowledge-based Policy Refinement Process



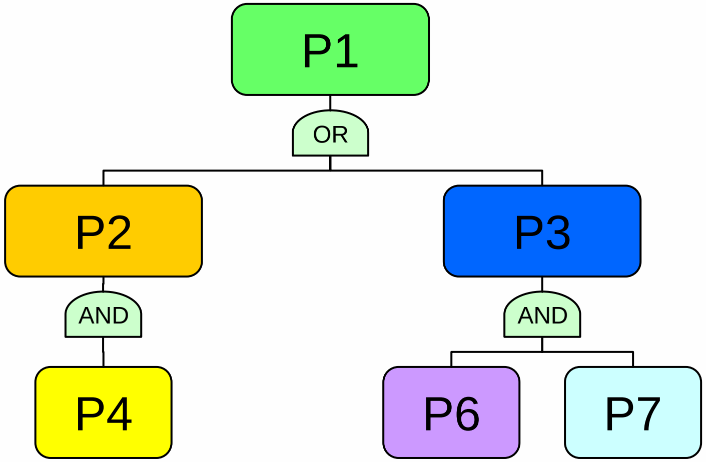
Policies represented as state labels

Policies represented as tree

An Overview of Expertise Knowledge-based Policy Refinement Process



Policies represented as state labels



Policies represented as tree

Model Checking

- Objective
  - Given a model *M* and a formula *f*, retrieve the execution path *σ*, which satisfies the formula *f*
  - Formally: $M, \sigma \models f$
- Model checking as pattern matching
  - Pattern context and problem as formula $\phi_{context}$ and $\phi_{problem}$
  - Workflow model *M*
  - Find any (finite) execution path π satisfying: $M, \pi \models \phi_{context} \wedge \phi_{problem}$
- Main obstacle
  - Both sets of atomic propositions use different vocabularies

$$
\begin{aligned}
M &: \langle W, R, L \rangle \\
W &: \text{a set of states} \\
R \subseteq W \times W &: \text{a set of state transitions} \\
L &: W \longrightarrow \mathcal{P}(AP), \text{ the state labeling function}
\end{aligned}
$$

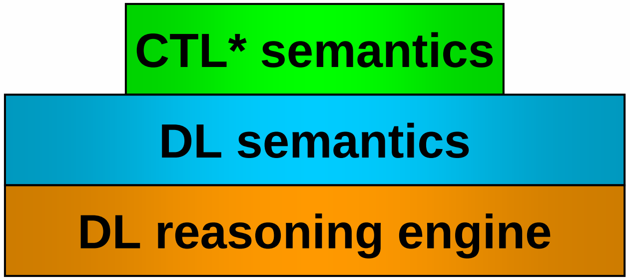
Kripke model M

$$
\begin{aligned}
\alpha &::= p \mid \neg\alpha \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha) \mid (\alpha\, \mathcal{U}\, \alpha) \mid \mathbf{X}\, \alpha \mid \mathbf{G}\, \alpha \mid \mathbf{F}\, \alpha \\
& \quad p \in \text{AP}
\end{aligned}
$$

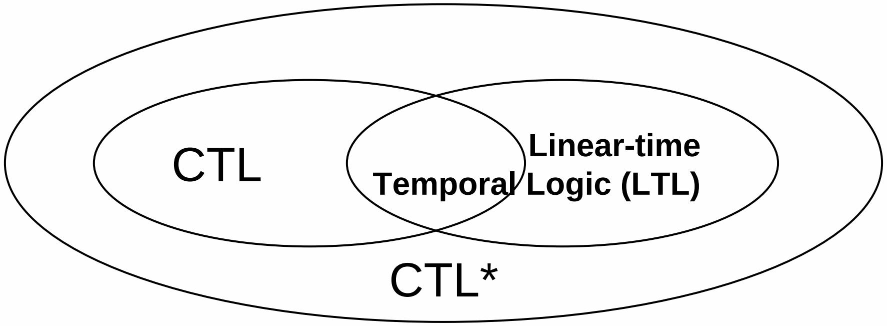
Syntax rule for constructing LTL formula

Description Logic-based Model Checking (I)

- Idea:
  - emulate the CTL* semantics on top of the Description Logic semantics
  - Use instance checking reasoning



- Approach
  - Define ontology of atomic propositions as a common vocabulary between *M* and *f*
  - Define CTL* semantics on top of description logic semantics
  - Represent *M* as individual (instance) assertions
  - Represent *f* as concepts (classes)
  - Perform instance checking



Description Logic-based Model Checking (II)

- Translated query:

$$M, \sigma_0 \models f \;?\; \Leftrightarrow: \;\; KB \models C(x) \;?$$

- Legend:
  - M : Kripke model
  - σ_0 : first state of the path
  - f : temporal logic formula
  - KB: knowledge base
  - C : concept representing f
  - x : instance representing σ_0

- Informally:
  - Does the path starting from state σ_0 of model M fulfill the formula f?

    $\Leftrightarrow$

  - Based on knowledge base KB, is the instance x a member of concept C?

Contributions

- Automated policy refinement process by using expertise knowledge
- Capturing the expertise knowledge using formalized patterns
  - Effectively capture domain experts' knowledge pertaining to workflow security (finance, healthcare, government, etc.)
  - The experts' knowledge can be directly used by the automated refinement process
- Description logic-based model checking
  - Enable model checking in heterogeneous environment (i.e. compliance check of web services behavior against customer policy)

End

Thank you!