Research @ RELEASeD

The RELEASeD Laboratory

Achieving Excellence in Software Development

released.info.ucl.ac.be

RELEASeD – the lab

REsearch Laboratory on
software Evolution And Software Development

- Computing Science Engineering Department (INGI)
- Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM)
- Université catholique de Louvain (UCL)

RELEASeD – the research

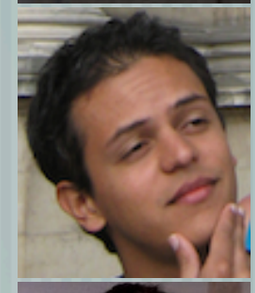
- Current research domains
  - Software Evolution Technology
  - Software Development Technology
  - Context-Oriented Programming
- Strong focus on tools, techniques and languages

RELEASeD – the team

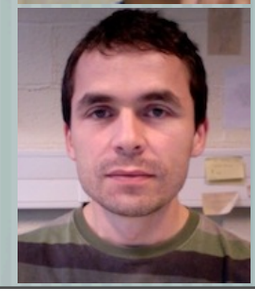
Kim Mens , Professor, Head of RELEASeD group
Kim.Mens@uclouvain.be

Sergio Castro , Teaching and Research Assistant
Sergio.Castro@uclouvain.be

Ángela Lozano , Post-Doctoral Researcher
Angela.Lozano@uclouvain.be



Nicolás Cardozo , PhD student
nicolas.cardozo@uclouvain.be



Sebastián González , Post-Doctoral Researcher
s.gonzalez@uclouvain.be

Some focused presentations

- Context-Oriented Programming [Sebastian, Nicolas, Kim]
  - Recent research on COP : Version Contexts , Context Traits & Context Petri Nets
- Software Quality Assurance [Angela, Kim]
  - Tools and techniques to improve the structural quality of software : Usage Contracts & Mendel
- Java-Prolog Interoperability [Sergio, Kim]

SQA @ RELEASeD

Kim Mens, Angela Lozano

Software Quality Assurance

- Mendel
  - a code recommendation tool that provides structural recommendations to software developers based on local structural properties in the source code of a system
- Usage Contracts
  - a unit testing-like approach to enable developers to express and verify structural coding conventions in a seamless way

COP @ RELEASeD

Kim Mens, Sebastian Gonzalez, Nicolas Cardozo

COP @ RELEASeD

- Research on COP since 10 years
  - Ambience [2008], Subjective-C [2010], SCOPJ [2011], Phenomenal Gem [2012]
- Most recent developments
  - Version Contexts [UCL 2013]
  - Context Traits [AOSD 2013]
  - Context Petri Nets [UCL 2013]

Context-Oriented Programming

- Motivation : Towards a Mindset Shift

- From:
  Programming in Isolation

forward!

- To:
  Programming with Context

Context-Oriented Programming

Motivation : Towards a Mindset Shift

From:

Programming in Isolation

To:

Programming with Context

Contexts

Situations of the surrounding execution environment to which specific behavior is associated

Context-Oriented Programming

Motivation : Towards a Mindset Shift

forward!

Caution

Design patterns

Plugin architectures

Conditional statements

...

Context-Oriented Programming

Conditional Statements : some variant of ...

```
m(a) {
        if (  ) { IE logic }
    else if (  ) { Opera logic }
    else if (  ) { Chrome logic }
    else if (  ) { Safari logic }
    else if (  ) { Firefox logic }
    else                                                             { default logic }
}
```

Context-Oriented Programming

Conditional Statements : some variant of ...

```
m(a) {
    if ( ) { IE logic }
    else if ( ) { Opera logic }
    else if ( ) { Chrome logic }
    else if ( ) { Saf
    else if ( ) { Fir
                  { def
```

Adaptable

Tangled
Scattered
Fixed
No reuse
Complex logic

Context-Oriented Programming

With design patterns ...

```
m(a) {
   strategy.m(a)
}
```

```
m(a) { IE strategy }
```

```
m(a) { Opera strategy }
```

```
m(a) { Chrome strategy }
```

```
m(a) { Safari strategy }
```

```
m(a) { Firefox strategy }
```

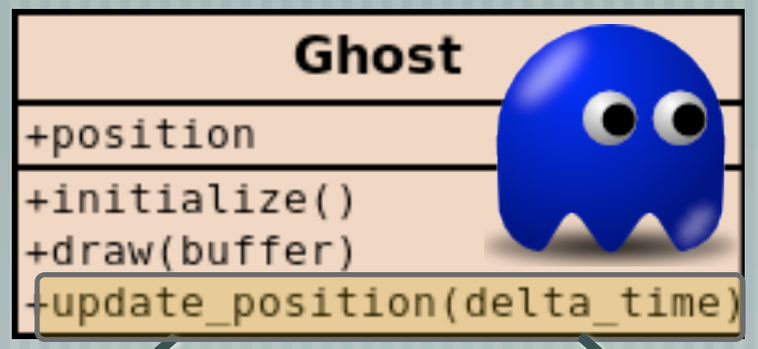
```
m(a) { default strategy }
```

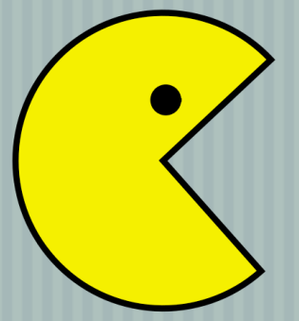
Modular
Open

Infrastructural burden
Anticipated adaptation points

Context-Oriented Programming

A Toy Example (really)





(default context)

adaptation

CODE: move toward pacman

Context Expert

adaptation

CODE: move randomly in the maze

Context Beginner

Context-Oriented Programming

An excerpt of the COP implementation of Pac-Man™ in Phenomenal Gem

Default behaviour

```
class Ghost
  def update_position
    # spec: move randomly in the maze
    # …
  end
end
```

```
ghost.update_position # move randomly
```

Context-Oriented Programming

An excerpt of the COP implementation of Pac-Man™ in Phenomenal Gem

```
class Ghost
  def update_position
    # spec: move randomly in the maze
    # …
  end
end

expert_context = Context.new
expert_context.add_adaptation(Ghost, :update_position, true,
  proc { # spec: move toward pacman }
})
ghost.update_position # move randomly
expert_context.activate
ghost.update_position # move toward pacman
```

Context-Oriented Programming

An excerpt of the COP implementation of Pac-Man™ in Phenomenal Gem

```
class Ghost
  def update_position
    # spec: move randomly in the maze
    # …
  end
end

expert_context = Context.new
expert_context.add_adaptation(Ghost, :update_position, true,
  proc { # spec: move toward pacman }
})
ghost.update_position # move randomly
expert_context.activate
ghost.update_position # move toward pacman
```

Explicit Contexts

Definition of Adaptations (even dynamically)

(De)activation of contexts

Dynamic Behaviour Adaptation

Context-Oriented Programming

- Some COP language extensions we have been working on ...
  - Ambience [CLOS]
  - Subjective-C [Objective-C]
  - SCopJ [Java]
  - Phenomenal Gem [Ruby]
  - Context Traits [JavaScript]
  - plus a small educational prototype in Smalltalk

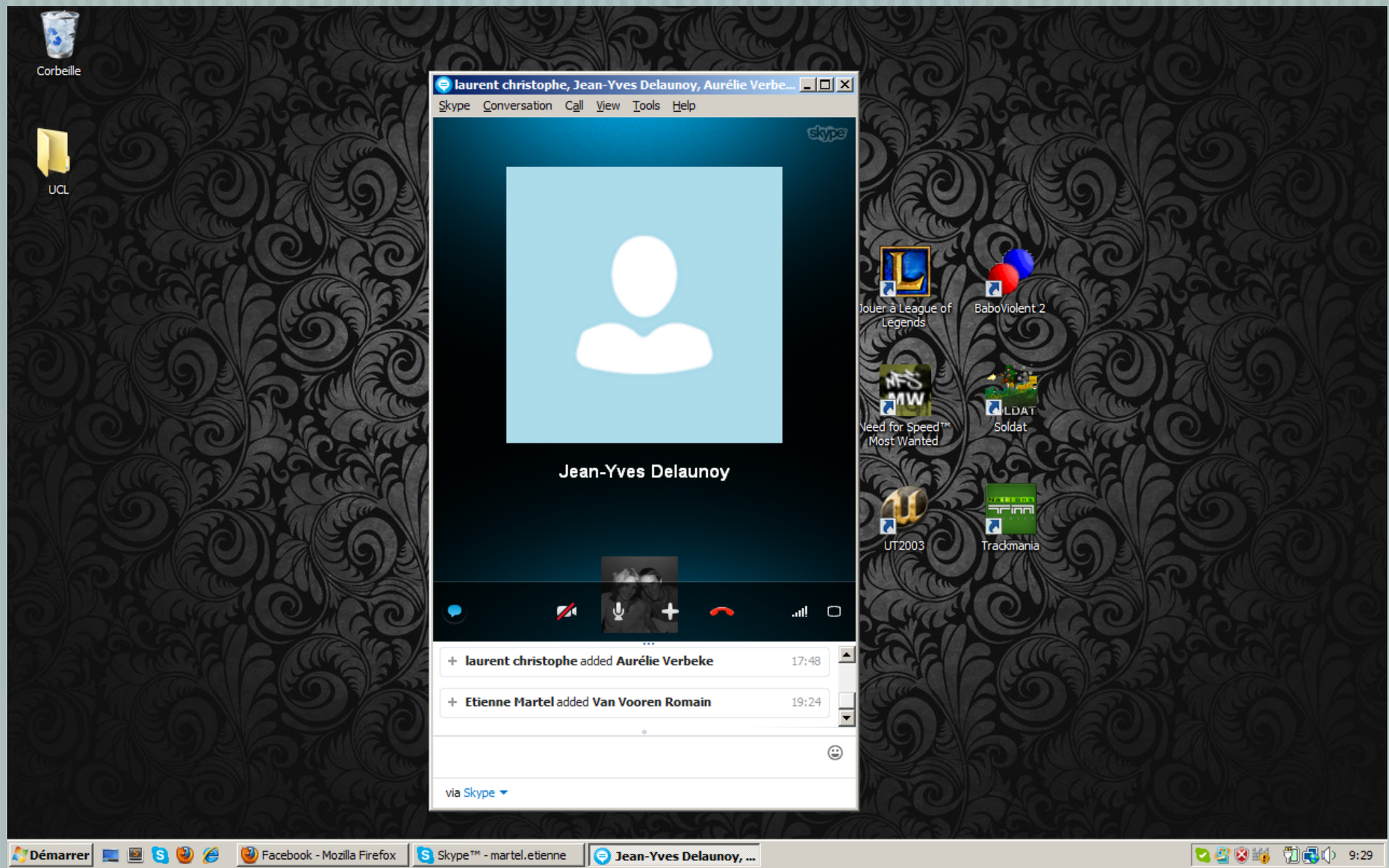
Version Contexts

Etienne Martel © June 2013 (MSc thesis, UCLouvain)
Advisors: Sebastian Gonzalez, Kim Mens

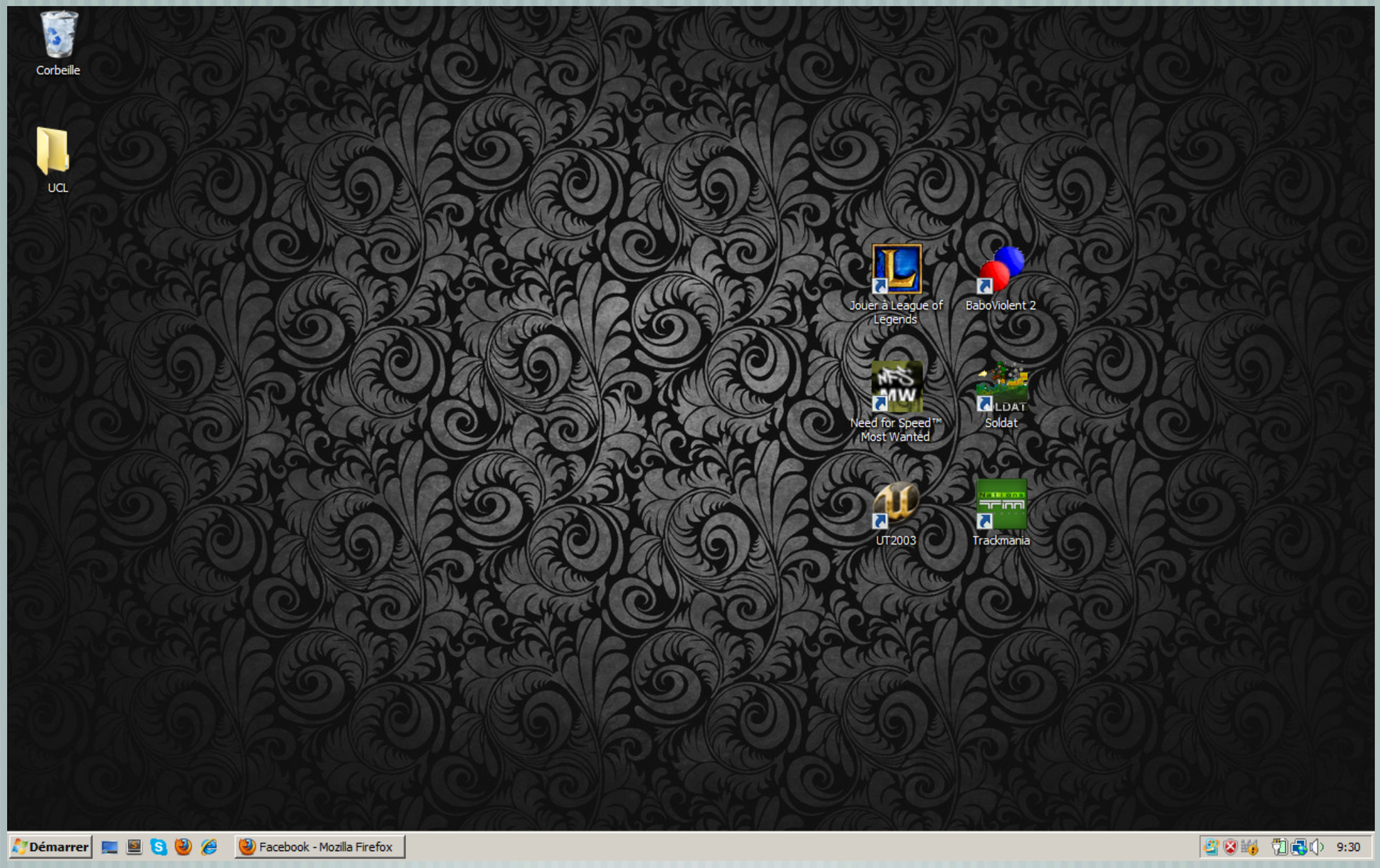
Motivation : Subversion

- Maintaining backward compatibility can lead to **software erosion**
  - e.g., in the source code of Subversion original functions like
    - `svn_client_checkout, svn_client_commit, svn_client_add, ...`
  - were subsequently extended with new functionality and parameters
    - `svn_client_checkout2,  svn_client_checkout3`
    - `svn_client_commit, svn_client_commit2, ..., svn_client_commit5`
    - `svn_client_add, svn_client_add2, ..., svn_client_add4`

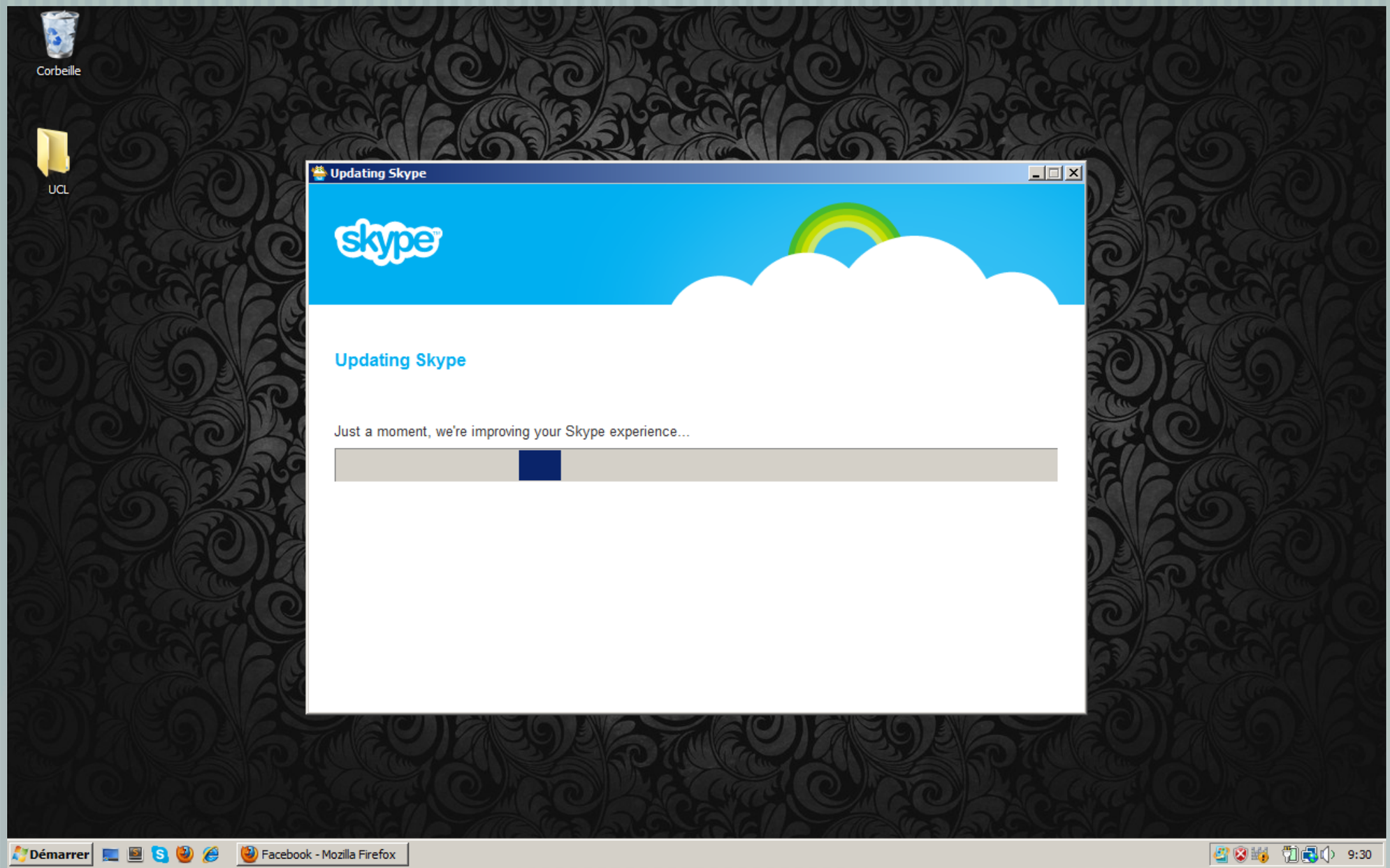
Motivation : Skype



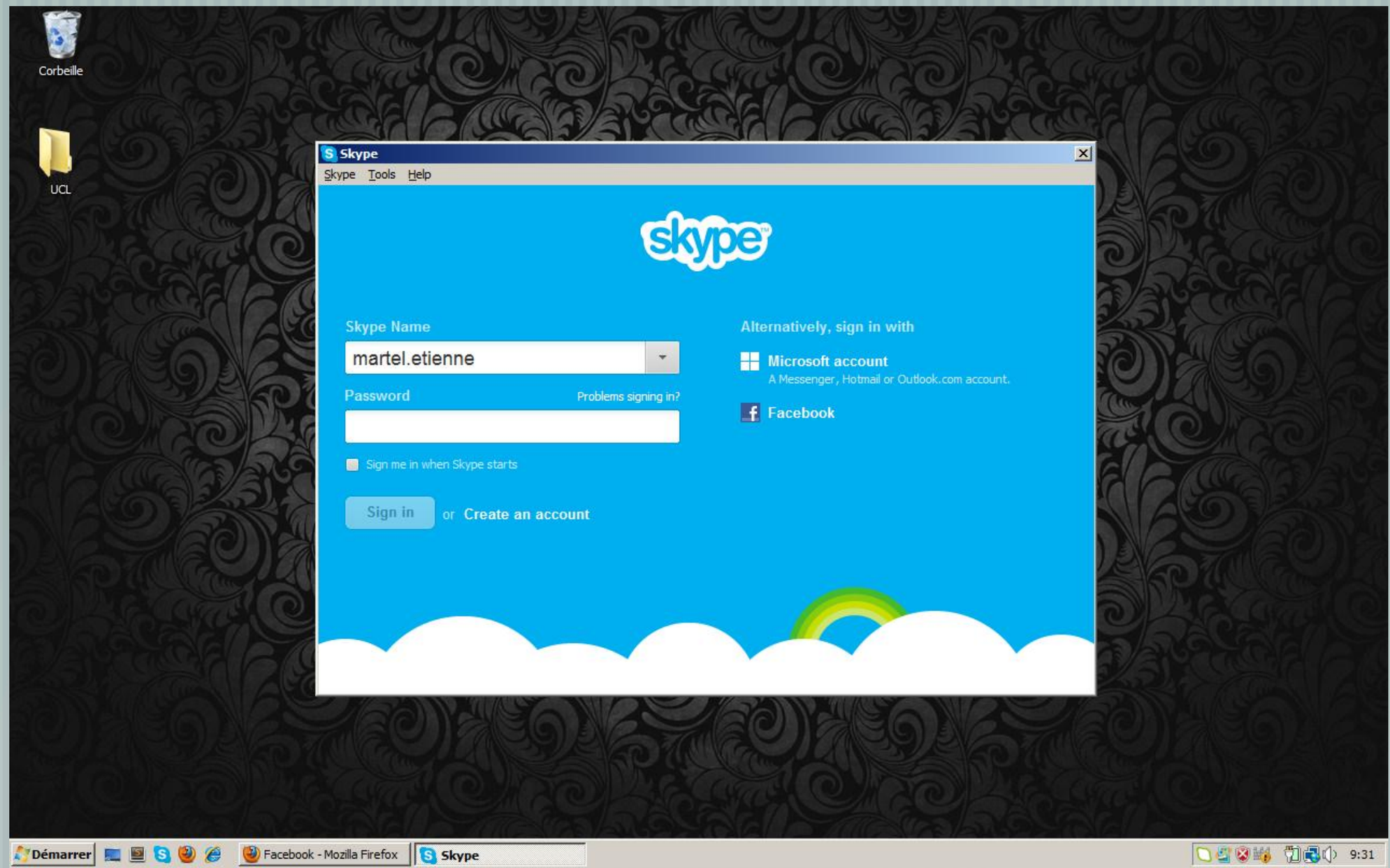
Motivation : Skype



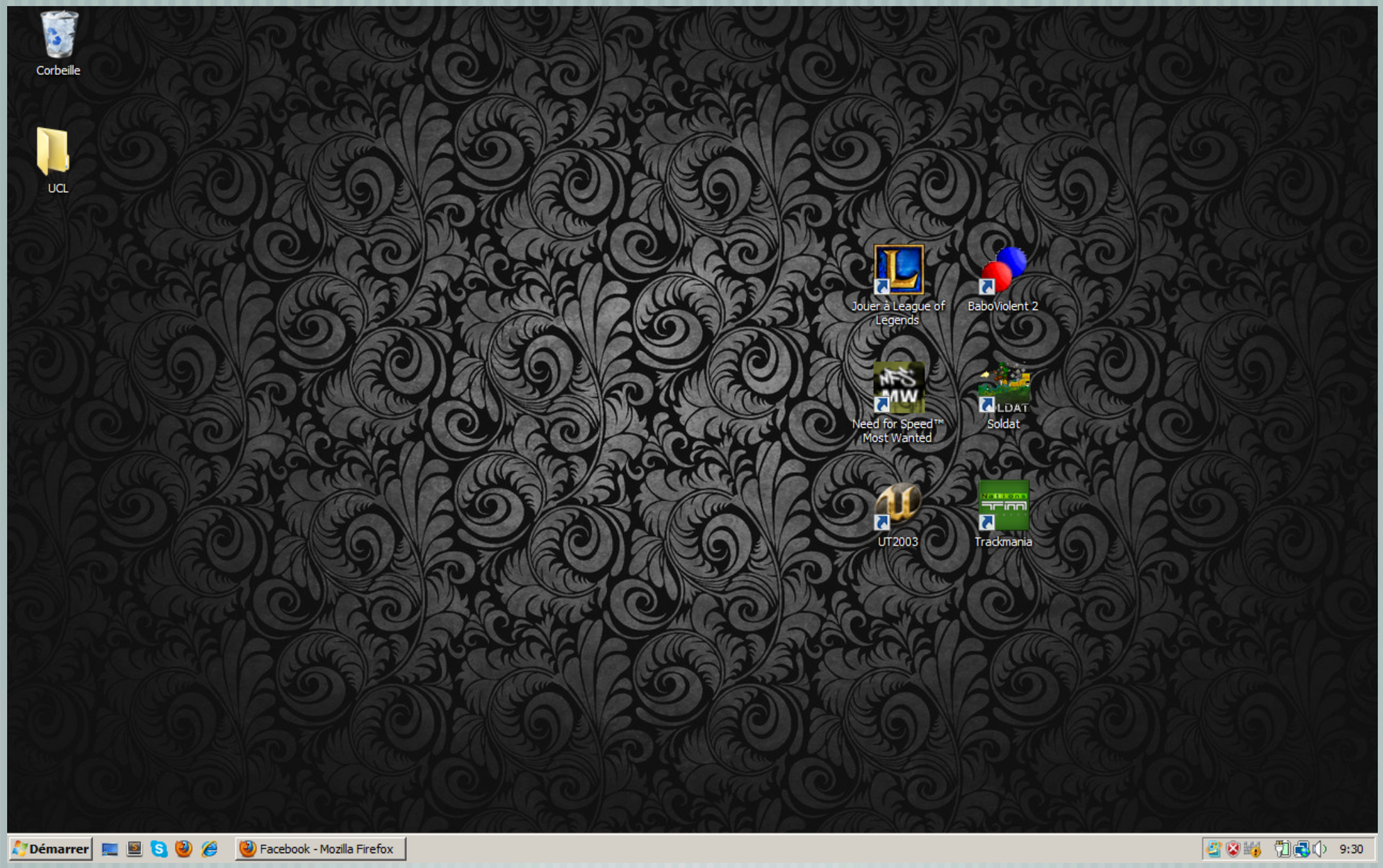
Motivation : Skype



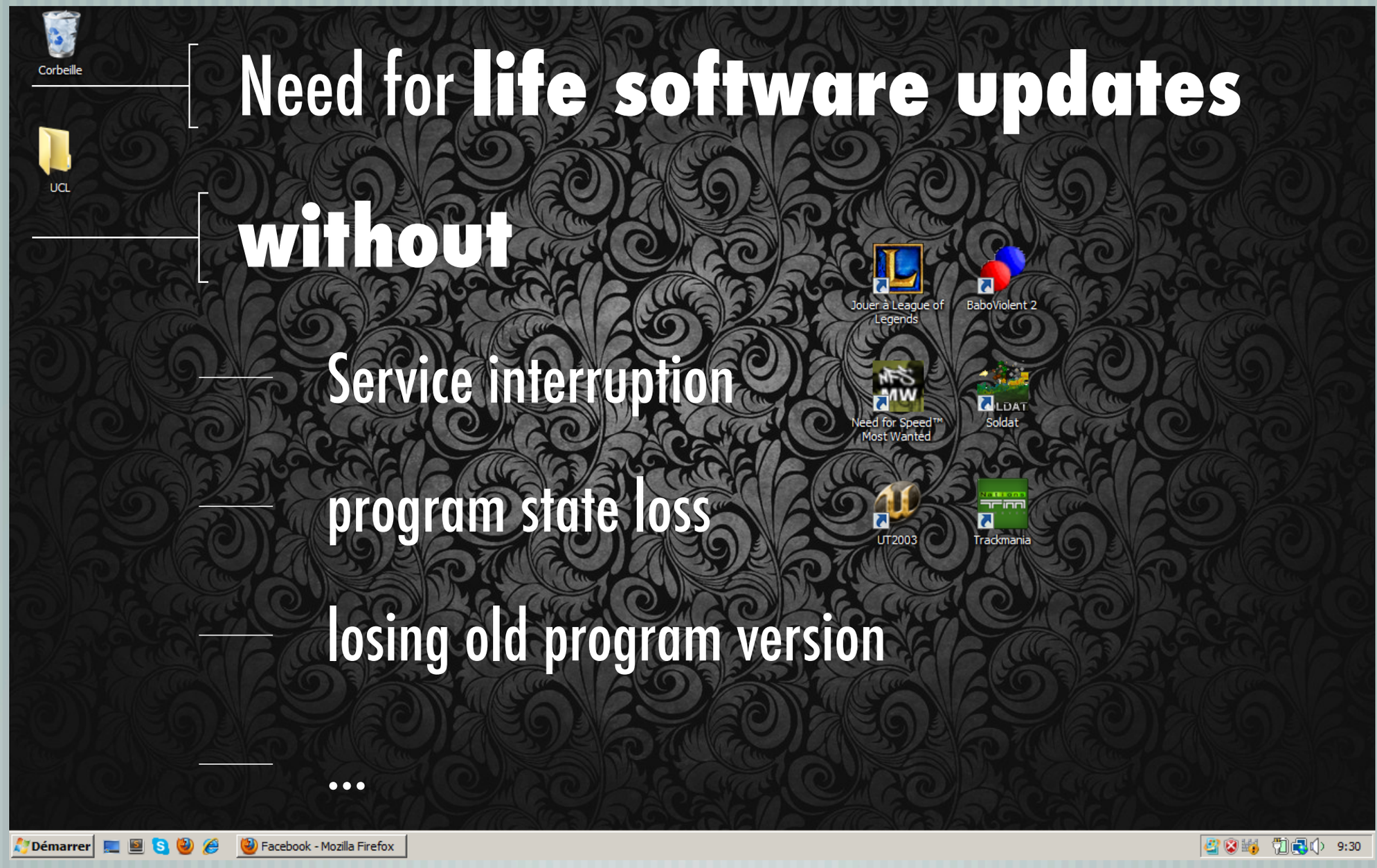
Motivation : Skype



Motivation : Skype



Motivation : Skype



Need for **life software updates**

without

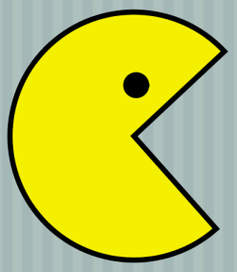
- Service interruption
- program state loss
- losing old program version
- ...

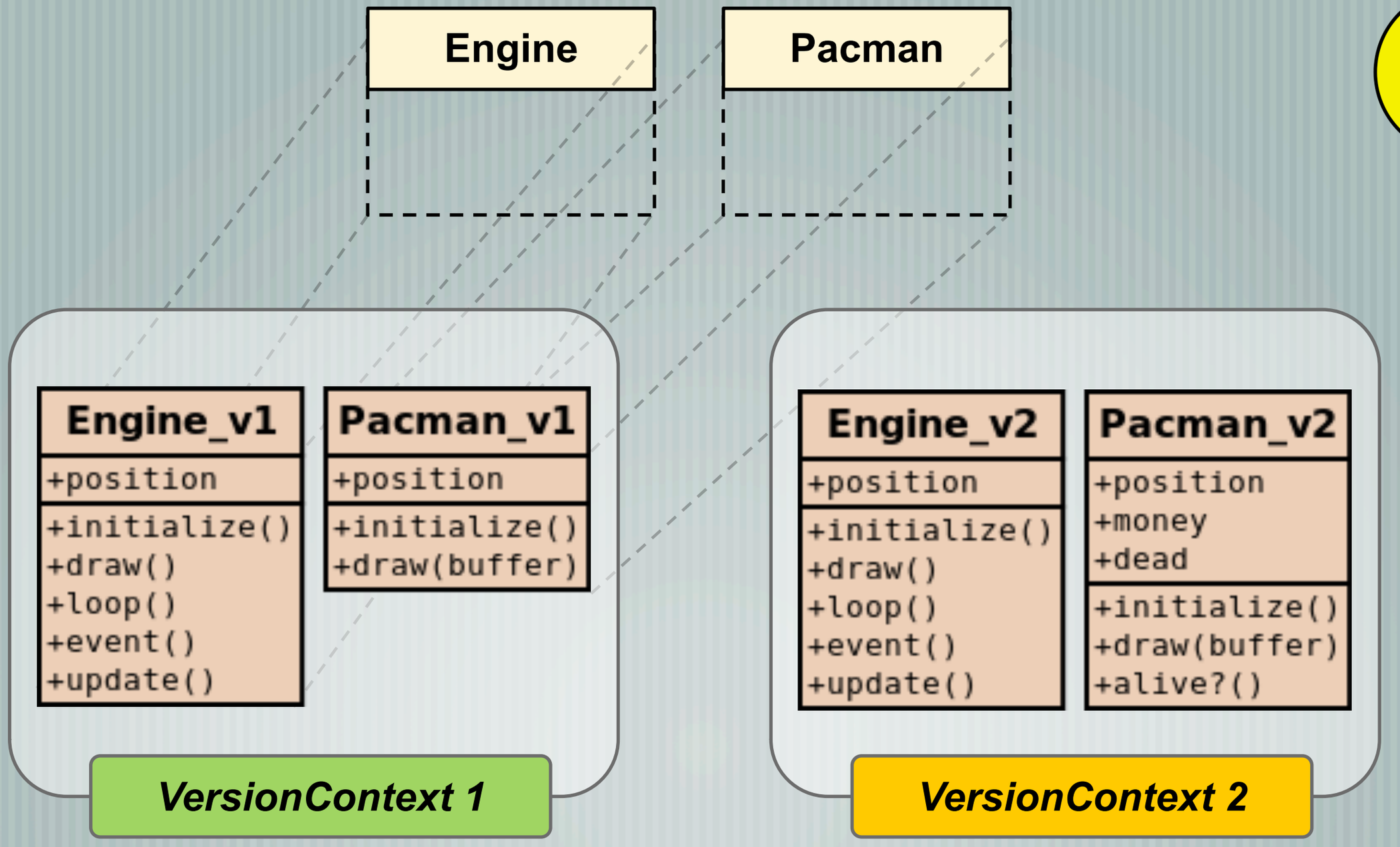
The idea

- Exploit COP by considering versions as a kind of context that can be dynamically (de)activated
  - activating that context = moving to a new version
  - deactivating = reverting to an older version
- Original use of COP in a way it hasn't been specifically conceived for

Version Contexts

Version Contexts extend Phenomenal Gem





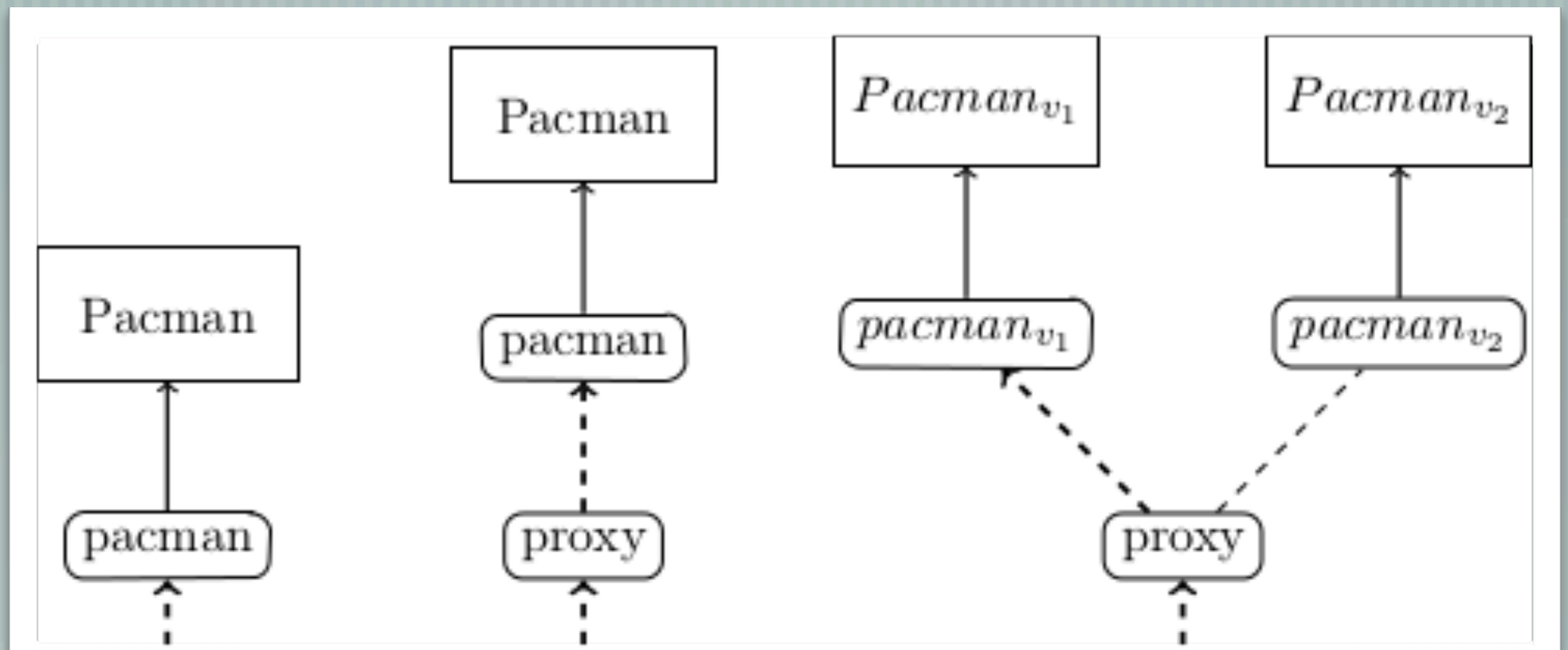
A Quick Demo

Implementation Issues

- Instance migration
  - migrating an existing instance to a new class
  - some capabilities are missing in Ruby
- Instance state
- Version mappings

Instance Migration

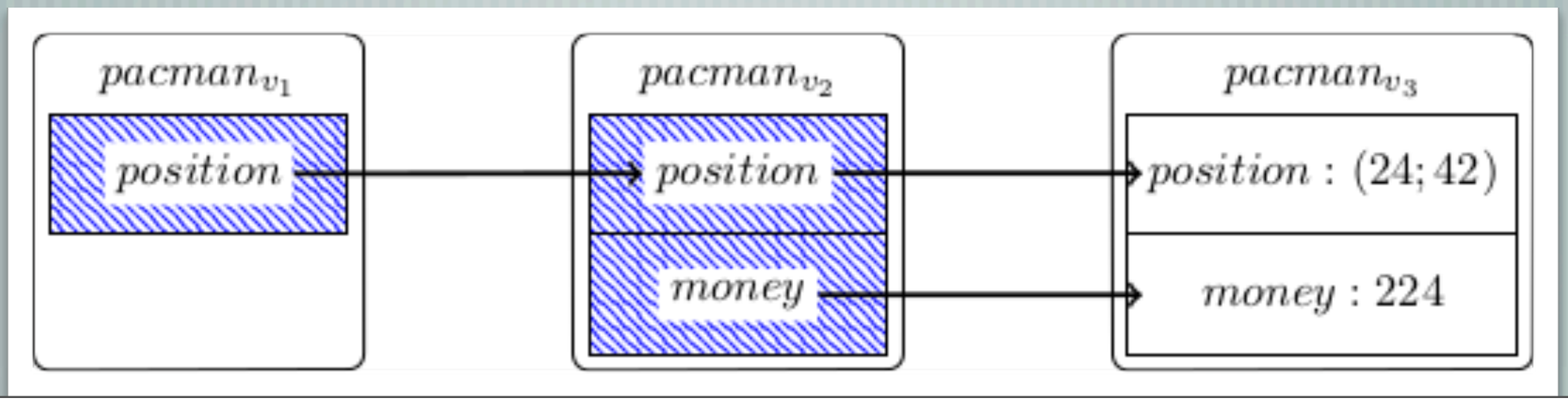
Object behavior must match the class definition of the currently active version context



TRADITIONAL NEW NEW WITH 2 CLASSES

Instance State

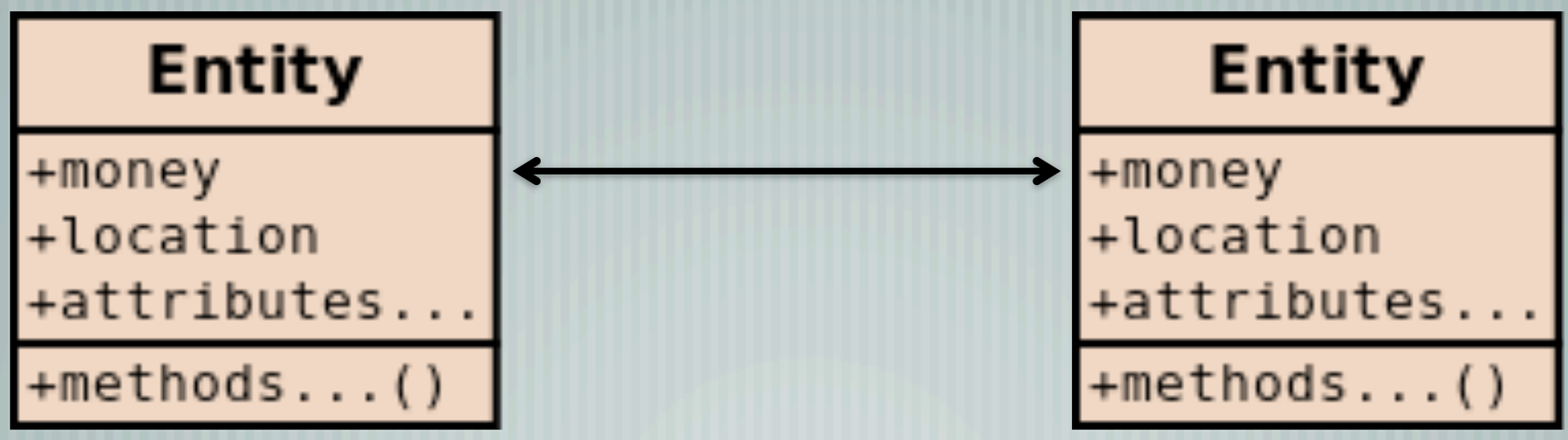
- Storing the values of attributes of each object for every version is inefficient.
- Solution :
  - store them for only one of the versions, e.g. the latest
  - redirect attribute access for the other versions



Version Mappings

- But what if the representation of an attribute evolves?
- For example, from money in $ to money in €

VERSION 1 **VERSION 2**

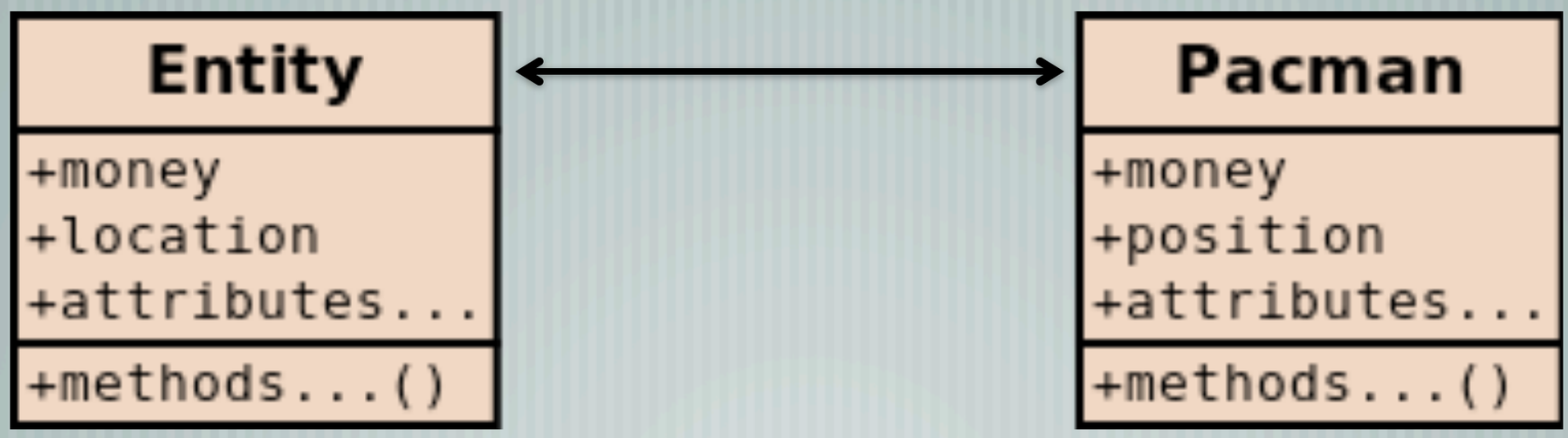


{ Entity@money in v1 is represented differently in v2 }

Version Mappings

Or what if a class is renamed from one version to another?

VERSION 1 **VERSION 2**

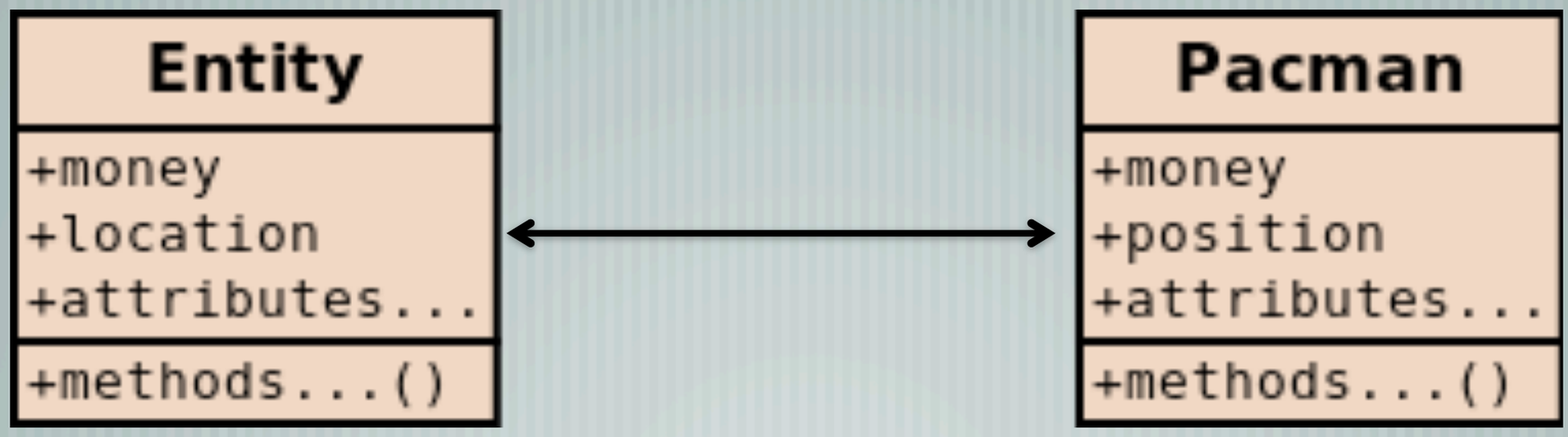


{ Entity in v1 = Pacman in v2 }

Version Mappings

Or what if an attribute gets renamed?

VERSION 1 **VERSION 2**



{ Entity@location in v1 = Pacman@position in v2 }

Version Mappings

- Need for version mappings to map from the old version to the new one (and back)
- (Bidirectional) functions defined by the programmer to render explicit these implicit changes between versions

```
class Pacman
  attr_accessor :money
  # 1 US$ = 27.435 FB
  result_mapping :@money, lambda {|val| val * 27.435 },
                          lambda {|val| val / 27.435 }
  # ...
end
```

Version Mappings

- Currently supported version mappings :
  - Name version mappings (classes, methods, attributes)
  - Parameter version mappings
  - Result version mappings
- More version mappings could (should) be added ...

Current Implementation Restrictions

- Linear versioning vs ~~branch versioning~~
- Granularity of an update: full version vs ~~delta version~~
- No support for concurrency

Context Traits

(very briefly)

Sebastian Gonzalez, Marius Colacioiu, Kim Mens, Walter Cazzola

The idea

- Dynamic Behaviour Adaptation Through Run-Time Trait Recomposition
- Paper presented at AOSD • Composition 2013
- Adopt the notion of Traits, a static composition mechanism, in a more dynamic setting, and use it as a building block for implementing COP

Context-Driven Trait Composition

T_1

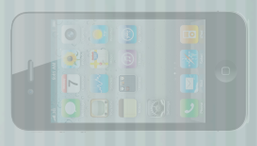
T_2

T_D

T_3

T_5

T_4



Context-Driven Trait Composition

T_2

T_1

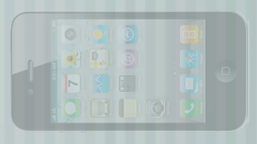
T_D

T_3

Default behaviour

T_5

T_4



Context-Driven Trait Composition

T_2

T_1

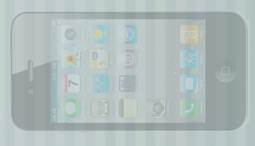
T_D

T_3

Context-specific adaptations

T_5

T_4



Context-Driven Trait Composition

T_2

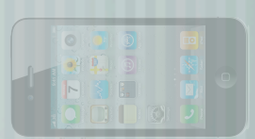
T_1

T_D

T_3

T_5

T_4



Context-Driven Trait Composition

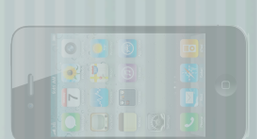
T_2

T_1

T_3

T_5

T_4



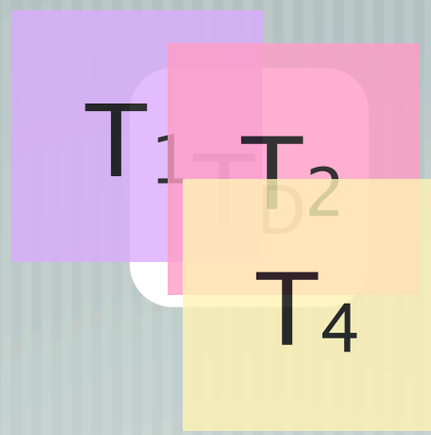
Context-Driven Trait Composition

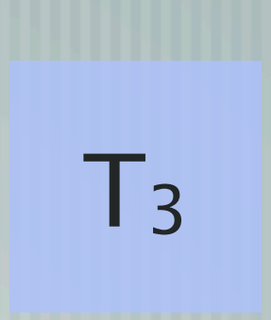
T_2

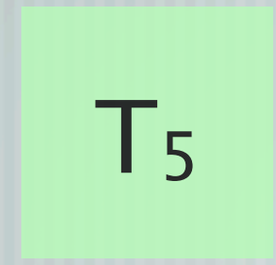
T_1 T_3 T_4

T_5

Context-Driven Trait Composition







Contributions

- Traits are convenient units of adaptation
  - more than single methods, less then full classes
- Support for flexible composition policies
  - e.g., non-linear (default policy = by activation age)
- Support for overriding behaviour with “proceed”
- JavaScript allows for easy definition of contexts, traits and composition
  - build on top op traits.js [Van Cutsem & Miller 2011]

In the paper...

- Contexts
- Traits
- Context-Driven Trait Compositions
- Composition Policies
- Behaviour Extensibility
- Context Traits in JavaScript
- Implementation Notes
- Case Studies
- Related Work
- Future Work

Context Petri Nets

Nicolas Cardozo © 2013

PhD thesis, advisors: Kim Mens & Theo D'Hondt

Research question

How to increase confidence in context-oriented systems that can dynamically adapt their behavior as context are being activated and deactivated ?

- Support for context dependency relations to express allowed and disallowed interactions between contexts
- Plus a mechanism to ensure consistency of the system with respect to those declared dependencies
  - both statically and dynamically

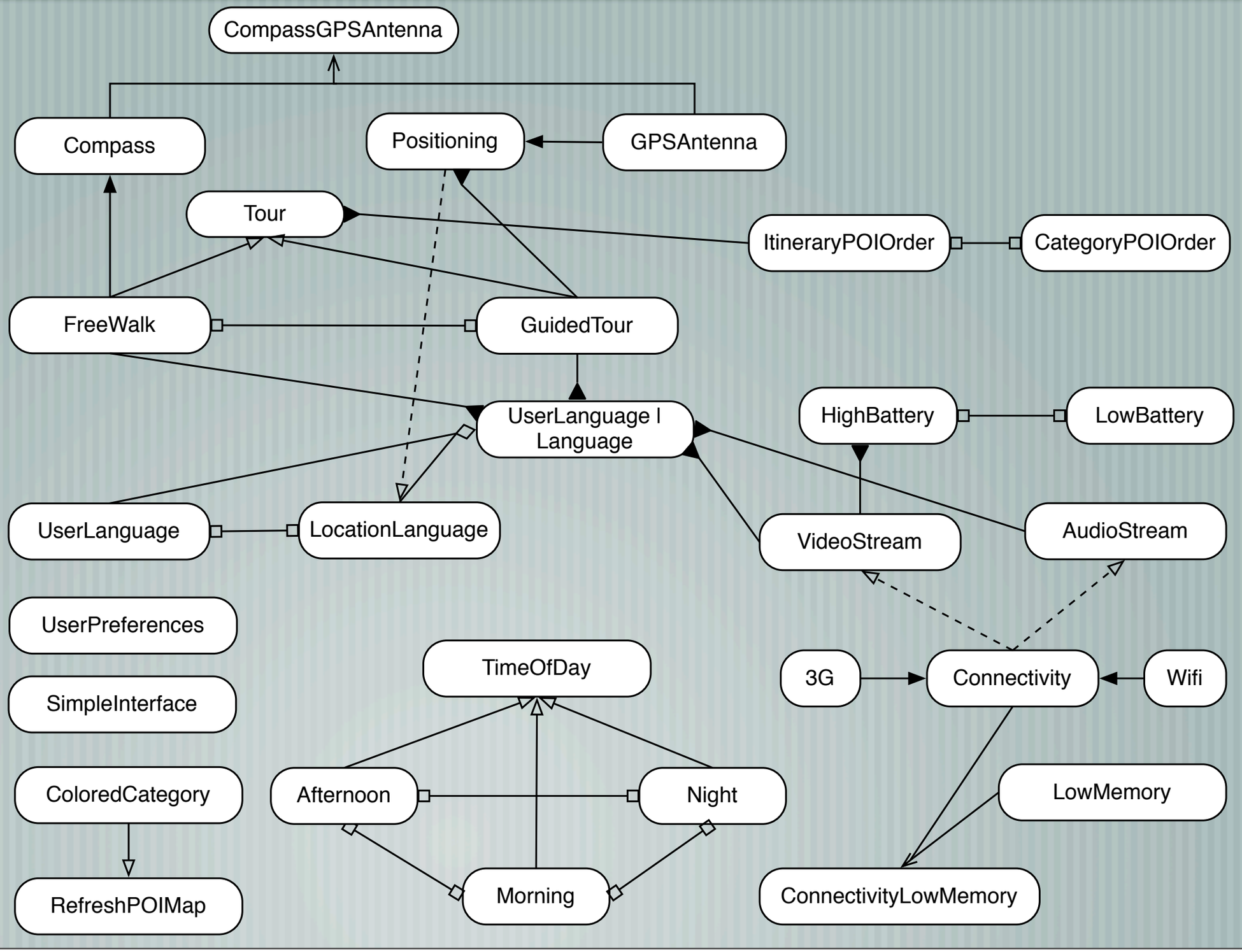
Context Dependency Relations

Context Dependency Relations

Context dependency relations

Different types :
- Exclusion
- Causality
- Implication
- Requirement
- Conjunction
- (Disjunction)
- (Suggestion)

Subjective-C



Research goal

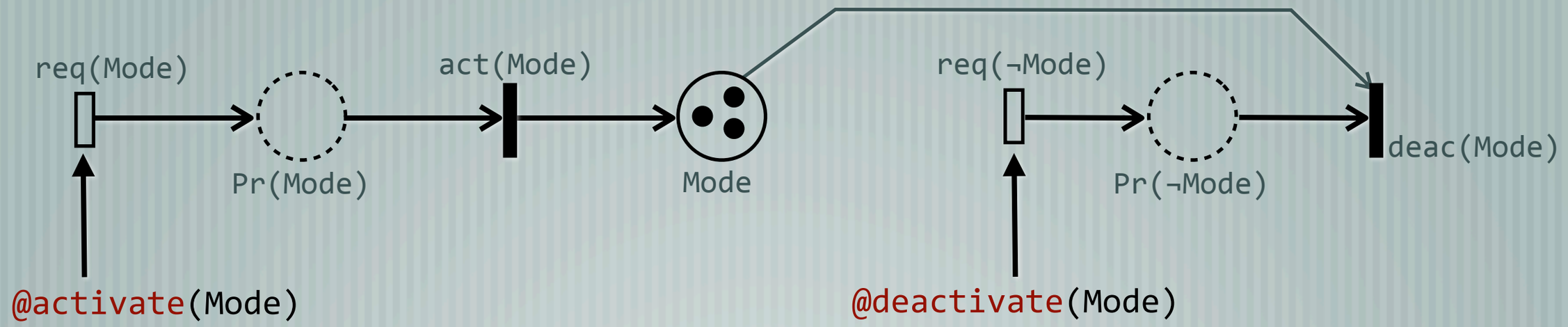
- How to ensure that
  - the declared model makes sense
  - verifying coherence and correctness of interactions
  - the model is respected by the system at run-time
  - ensure run-time consistency

Context Petri nets


Subjective-C

Singleton CoPN

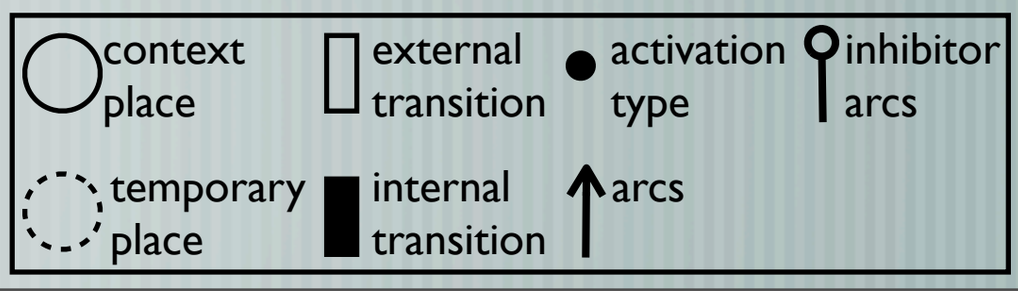
A context is represented by a **context Petri net (CoPN)** defined as $\mathcal{C} = \langle P_c, P_t, T_e, T_i, f, f_\circ, \rho, \mathcal{L}, m_0, \Sigma \rangle$




Subjective-C: Bringing Context to Mobile Platform Programming. *González, Sebastián and Cardozo, Nicolás and Mens, Kim and Cádiz, Alfredo and Libbrecht, Jean-Cristophe and Goffaux, Julien. International Conference on Software Language Engineering (SLE'10).*


Modeling and Analyzing Self-adaptive Systems With Context Petri Nets. *Cardozo, Nicolás and González, Sebastián and Mens, Kim and Ragnhild Van Der Straeten and D'Hondt, Theo. International Symposium on Theoretical Aspects of Software Engineering (TASE '13).*


Context Petri nets.
http://released.info.ucl.ac.be/Tools/Context-PetriNets



Context Dependency Relations

$\langle Q, \mathcal{C}_\mathrm{A}, \mathcal{C}_\mathrm{B} \rangle$ = [**addRequirementTo:** Tour **of:** Pos]

$\langle C, \mathcal{C}_\mathrm{A}, \mathcal{C}_\mathrm{B} \rangle$ = [**addCausalityFrom:** A **to:** B]

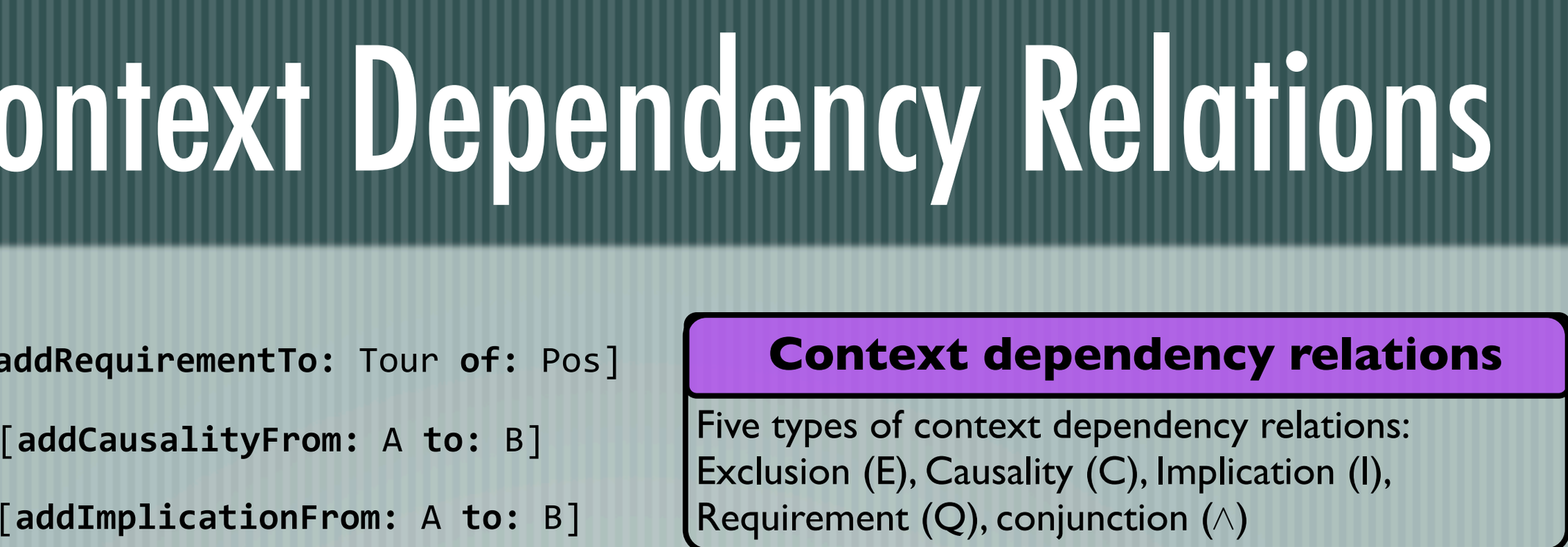
$\langle I, \mathcal{C}_\mathrm{A}, \mathcal{C}_\mathrm{B} \rangle$ = [**addImplicationFrom:** A **to:** B]

$\langle \wedge, \mathcal{C}_{\mathrm{A}_1}, \ldots, \mathcal{C}_{\mathrm{A}_n} \rangle$= [**addConjunctionOf:** [A_1, ..., A_n]]

$\langle E, \mathcal{C}_\mathrm{A}, \mathcal{C}_\mathrm{B} \rangle$ = [**addExclusionBetween:** Mode **and:** Pref]

Context dependency relations

Five types of context dependency relations: Exclusion (E), Causality (C), Implication (I), Requirement (Q), conjunction ($\wedge$)



[Subjective-C: Bringing Context to Mobile Platform Programming. *SLE*'10]

[Modeling and analyzing self-adaptive systems with context Petri nets. *TASE*'13]

Context Dependency Relations

$\langle Q, \mathcal{C}_A, \mathcal{C}_B \rangle$ = [**addRequirementTo:** Tour **of:** Pos]

$\langle C, \mathcal{C}_A, \mathcal{C}_B \rangle$ = [**addCausalityFrom:** A **to:** B]

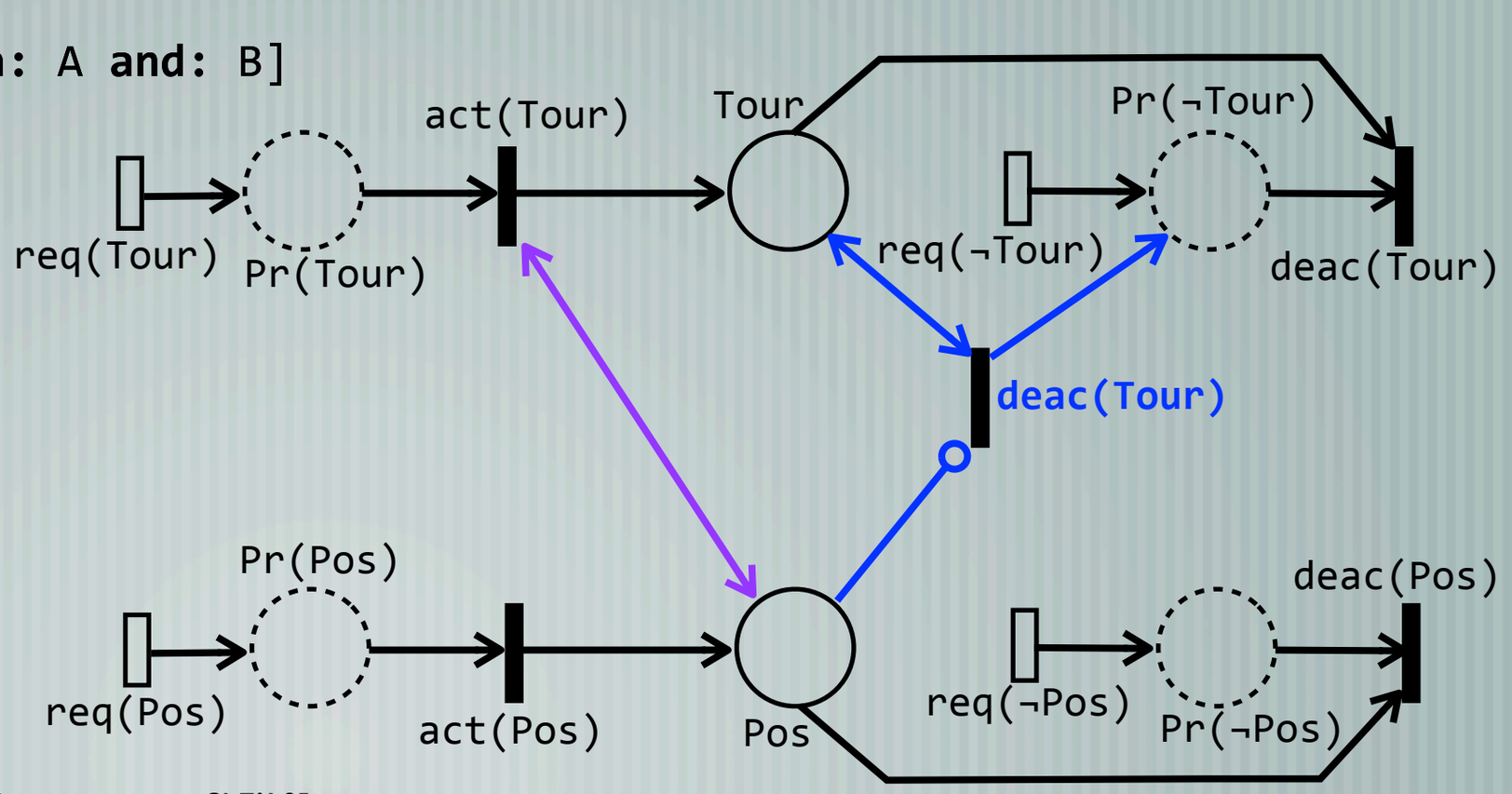
$\langle I, \mathcal{C}_A, \mathcal{C}_B \rangle$ = [**addImplicationFrom:** A **to:** B]

$\langle \wedge, \mathcal{C}_{A_1}, \ldots, \mathcal{C}_{A_n} \rangle$ = [**addConjunctionOf:** [A_1, ..., A_n]]

$\langle E, \mathcal{C}_A, \mathcal{C}_B \rangle$ = [**addExclusionBetween:** A **and:** B]

Context dependency relations

Five types of context dependency relations: Exclusion (E), Causality (C), Implication (I), Requirement (Q), conjunction ($\wedge$)



[Subjective-C: Bringing Context to Mobile Platform Programming. *SLE*'10]

[Modeling and analyzing self-adaptive systems with context Petri nets. *TASE*'13]

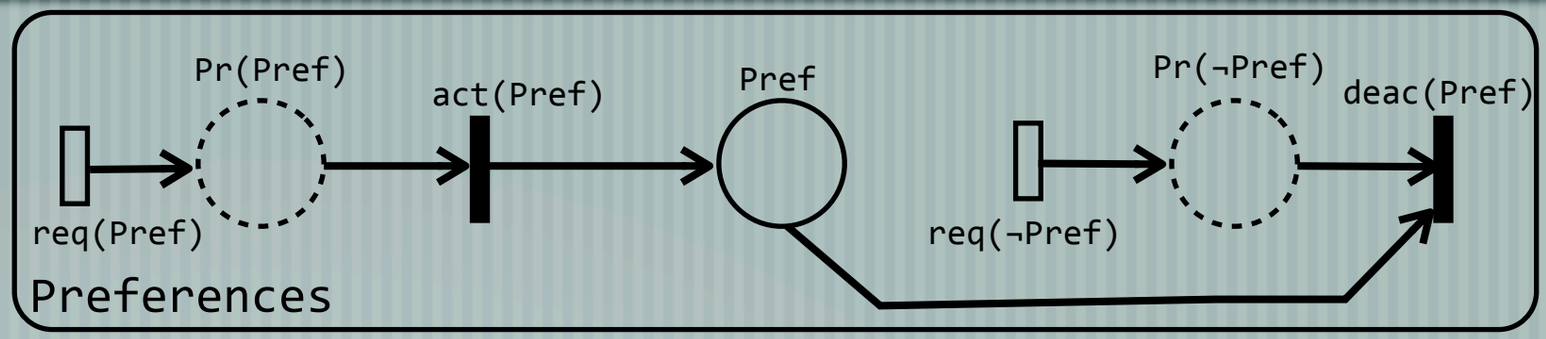
Composition of CoPNs

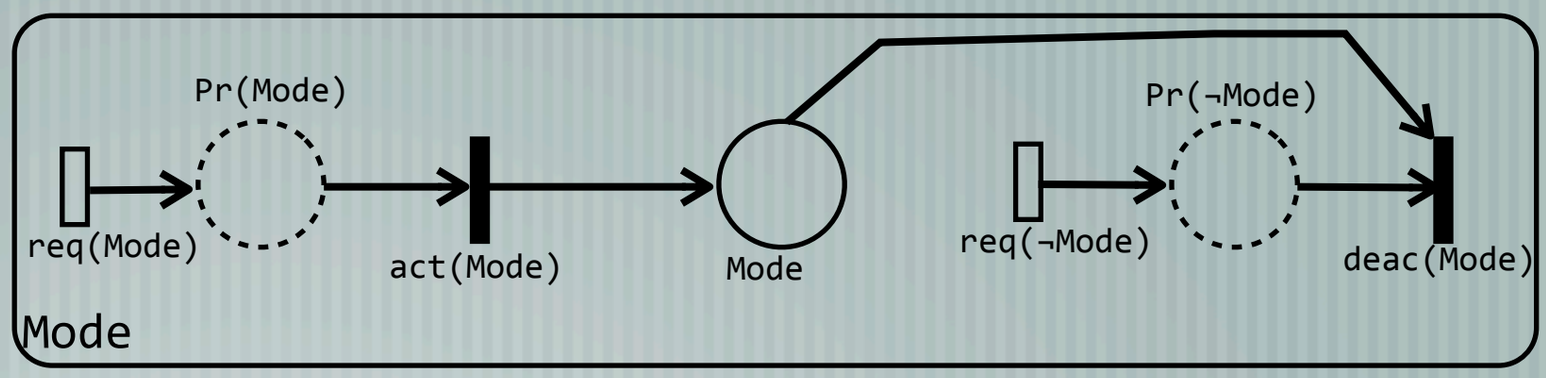
Composition

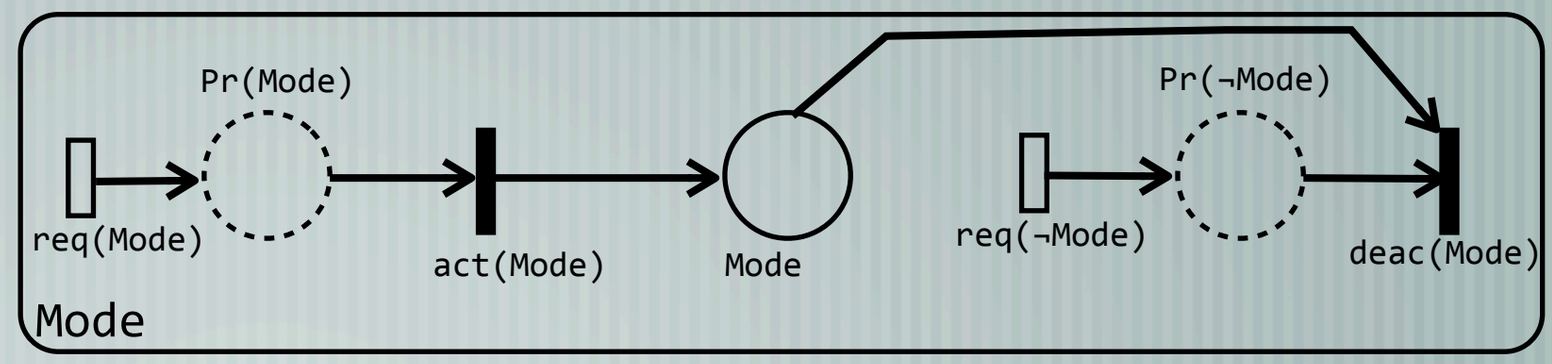
Given a set of contexts $\mathcal{S}$ and a set of context dependency relations $\mathcal{R}$ the composition is defined as :

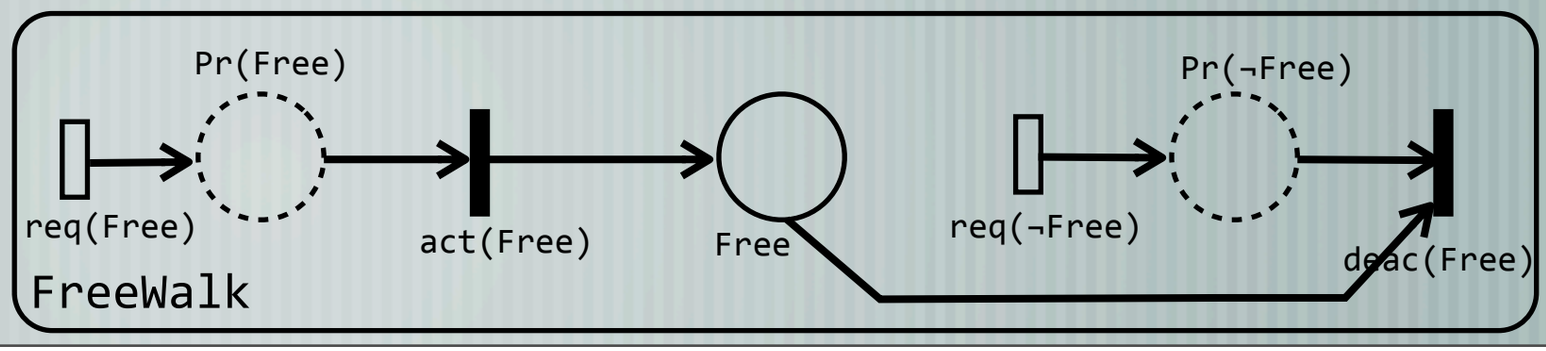
$\texttt{cons}(\texttt{ext}(\texttt{union}(\mathcal{S}), \mathcal{R}), \mathcal{R})$

- $\mathcal{S}$: contexts Preferences, Mode, FreeWalk
- $\mathcal{R}$: Exclusion (E) and causality (C) relations









Composition of CoPNs

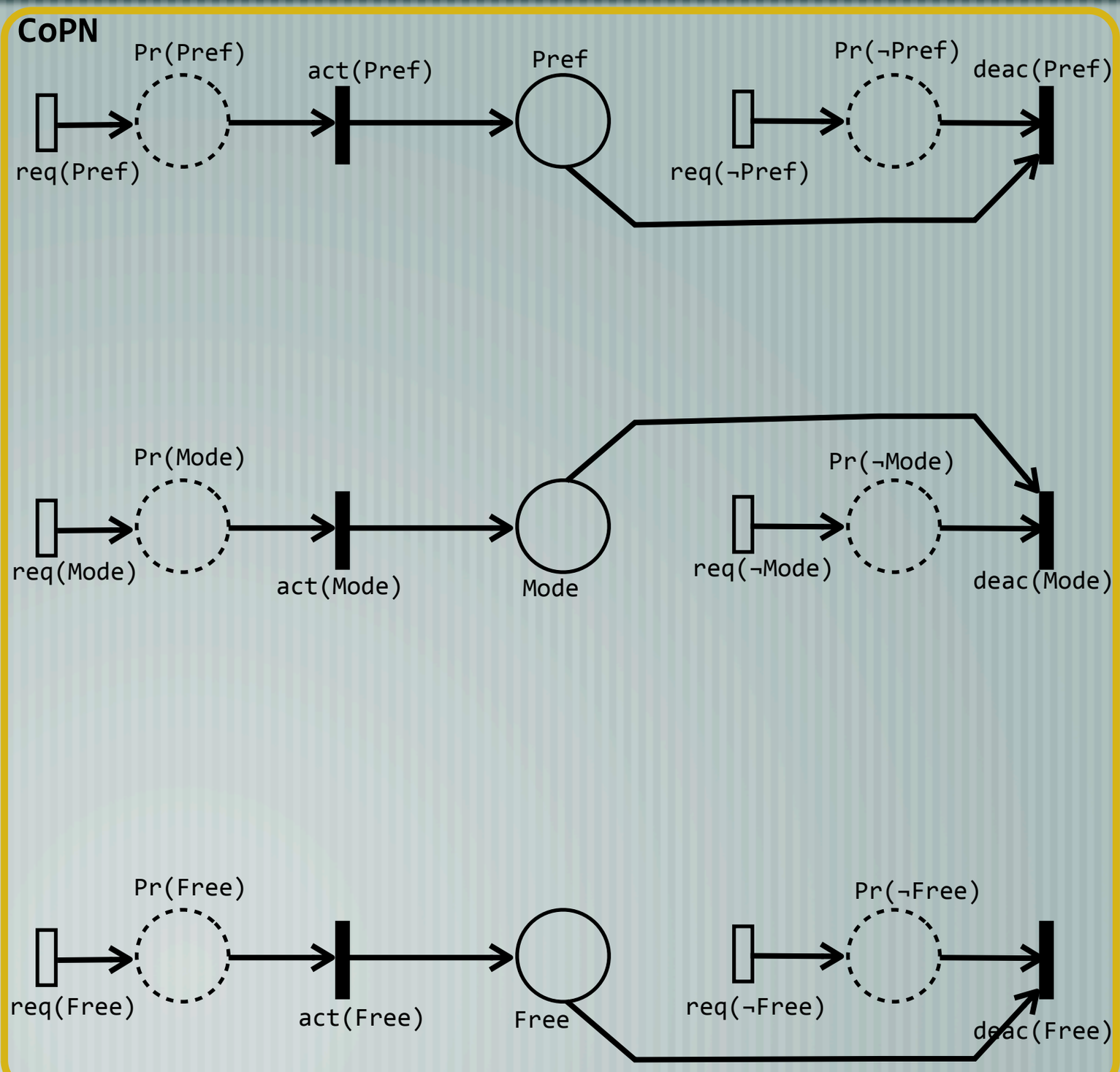
Composition

Given a set of contexts $\mathcal{S}$ and a set of context dependency relations $\mathcal{R}$ the composition is defined as :

$\texttt{cons}(\texttt{ext}(\texttt{union}(\mathcal{S}), \mathcal{R}), \mathcal{R})$

- $\mathcal{S}$: contexts `Preferences`, `Mode`, `FreeWalk`
- $\mathcal{R}$: Exclusion (E) and causality (C) relations

$\texttt{union}(\mathcal{S})$



Composition of CoPNs

Composition

Given a set of contexts $\mathcal{S}$ and a set of context dependency relations $\mathcal{R}$ the composition is defined as :

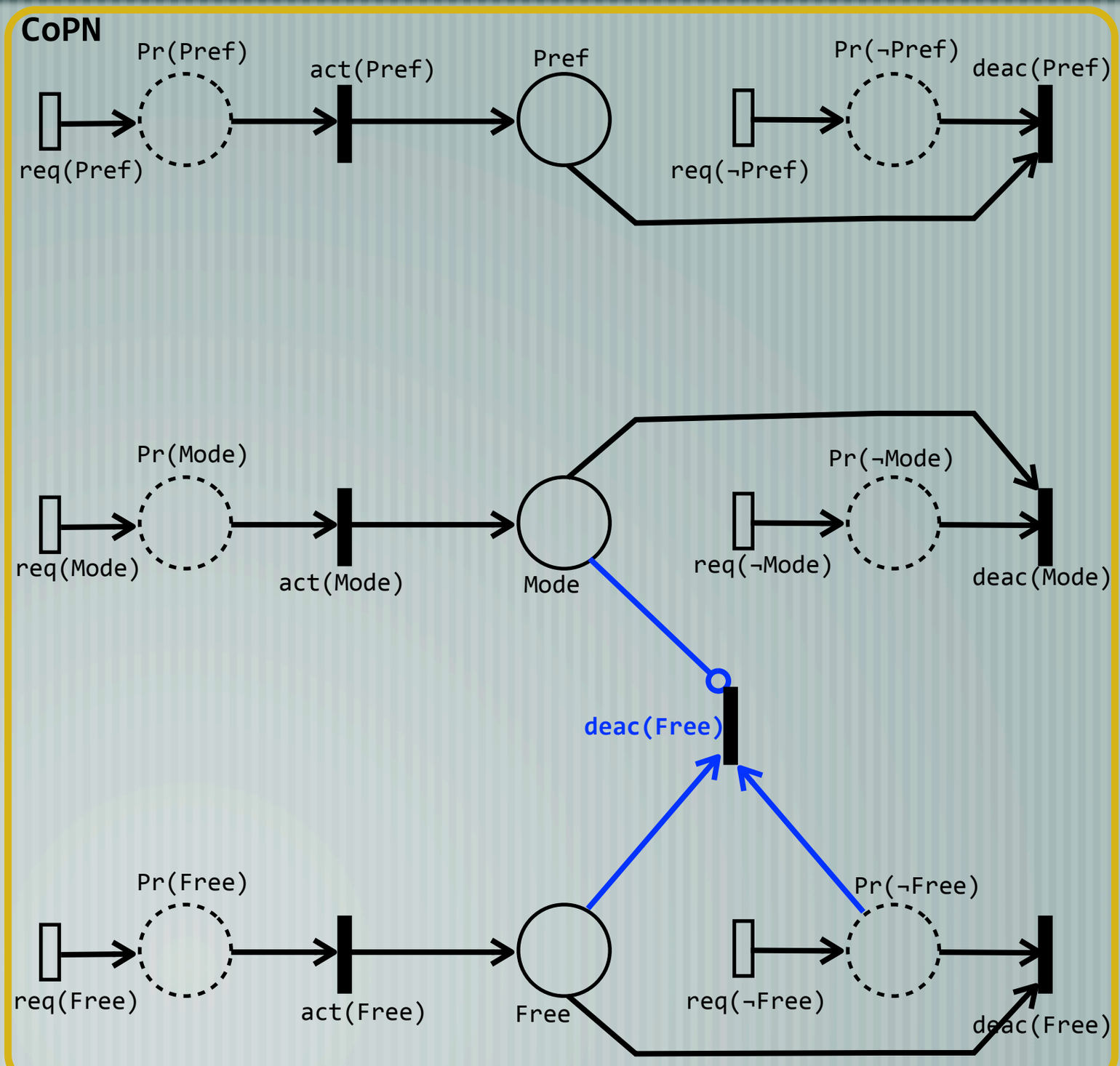
$\texttt{cons}(\texttt{ext}(\texttt{union}(\mathcal{S}), \mathcal{R}), \mathcal{R})$

- $\mathcal{S}$: contexts `Preferences`, `Mode`, `FreeWalk`
- $\mathcal{R}$: Exclusion (E) and causality (C) relations

$\texttt{union}(\mathcal{S})$

$\texttt{ext}(\mathcal{S}, \mathcal{R})$

- Add constraints specific to each context dependency relation



Composition of CoPNs

Composition

Given a set of contexts $\mathcal{S}$ and a set of context dependency relations $\mathcal{R}$ the composition is defined as :

$\texttt{cons}(\texttt{ext}(\texttt{union}(\mathcal{S}), \mathcal{R}), \mathcal{R})$

- $\mathcal{S}$: contexts `Preferences`, `Mode`, `FreeWalk`
- $\mathcal{R}$: Exclusion (E) and causality (C) relations

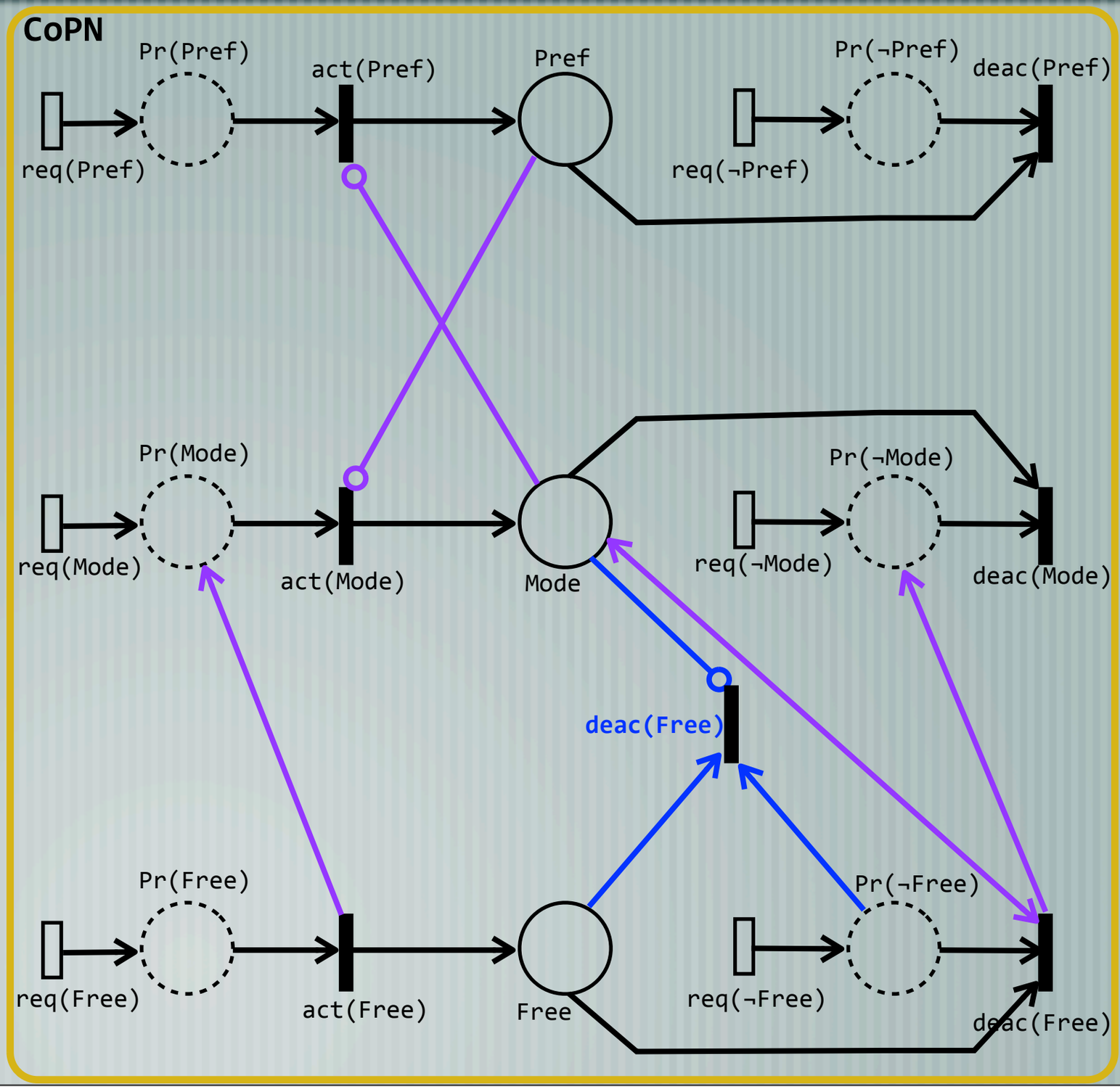
$\texttt{union}(\mathcal{S})$

$\texttt{ext}(\mathcal{S}, \mathcal{R})$

- Add constraints specific to each context dependency relation

$\texttt{cons}(\mathcal{S}, \mathcal{R})$

- Add constraints applicable for all defined contexts

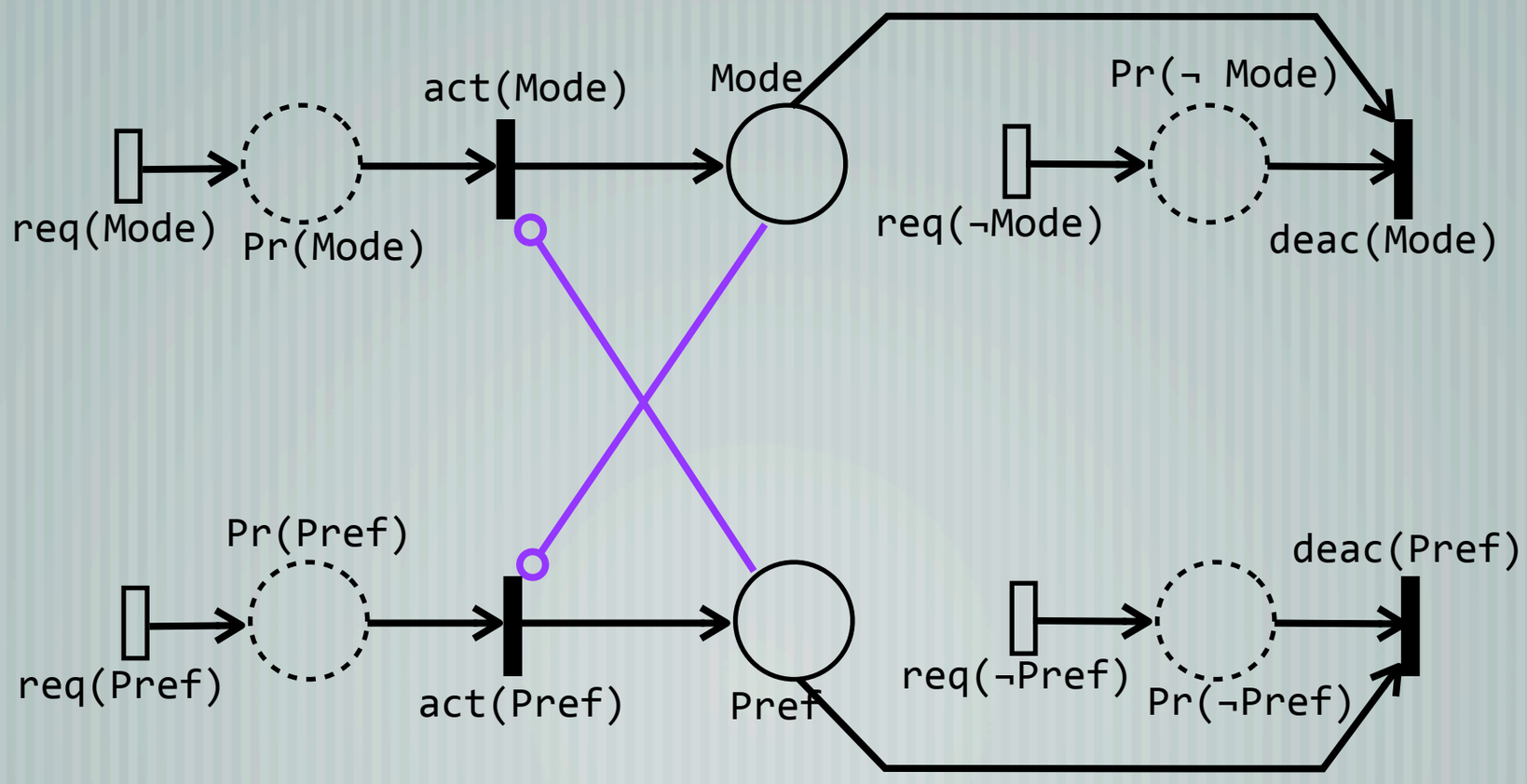


Consistent States

Consistent States

No enabled internal transitions, and no marked temporary places

```
[addExclusionBetween: Mode and: Pref]
```

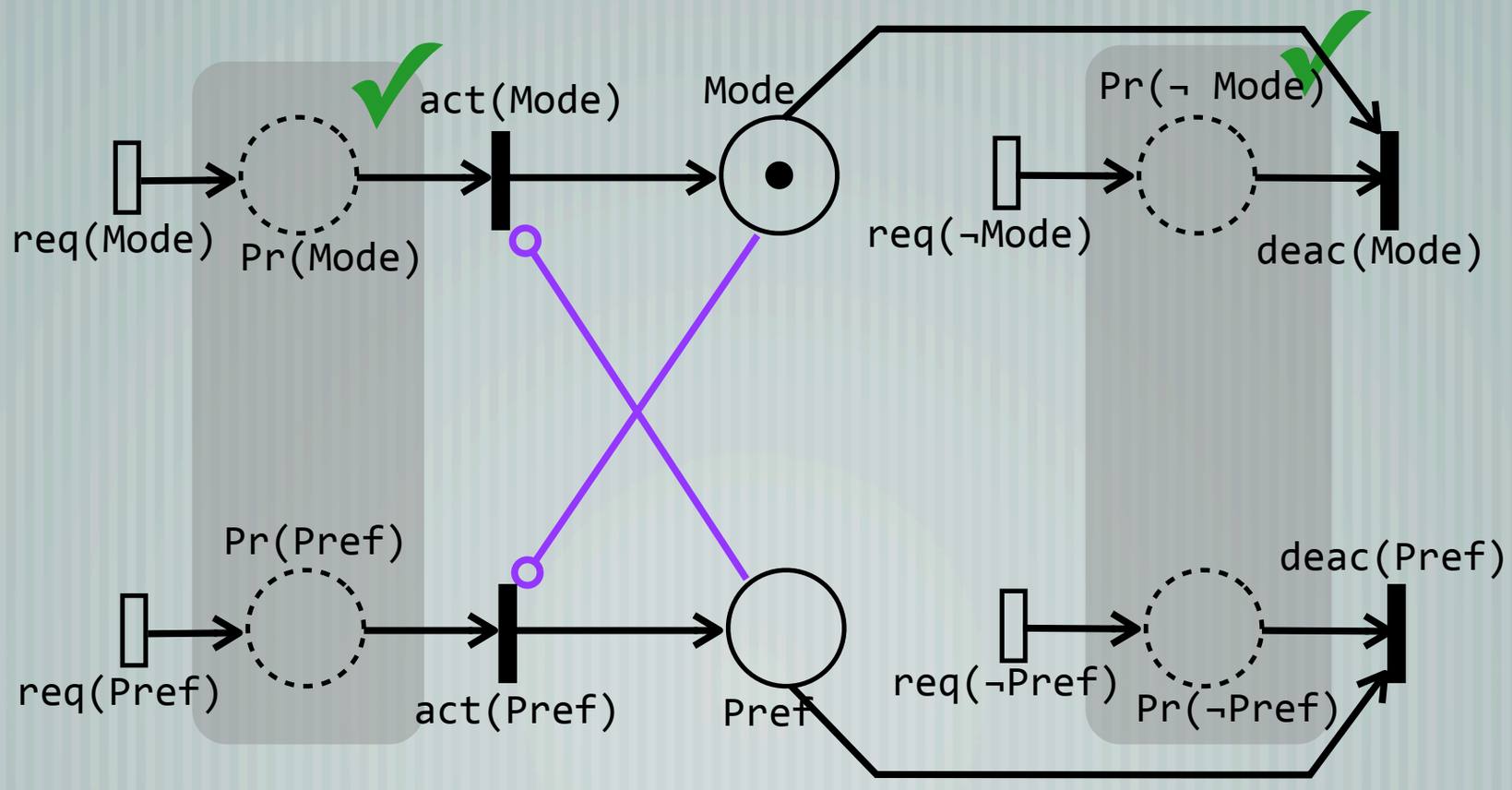


Consistent States

Consistent States

No enabled internal transitions, and no marked temporary places

```
[addExclusionBetween: Mode and: Pref]
```

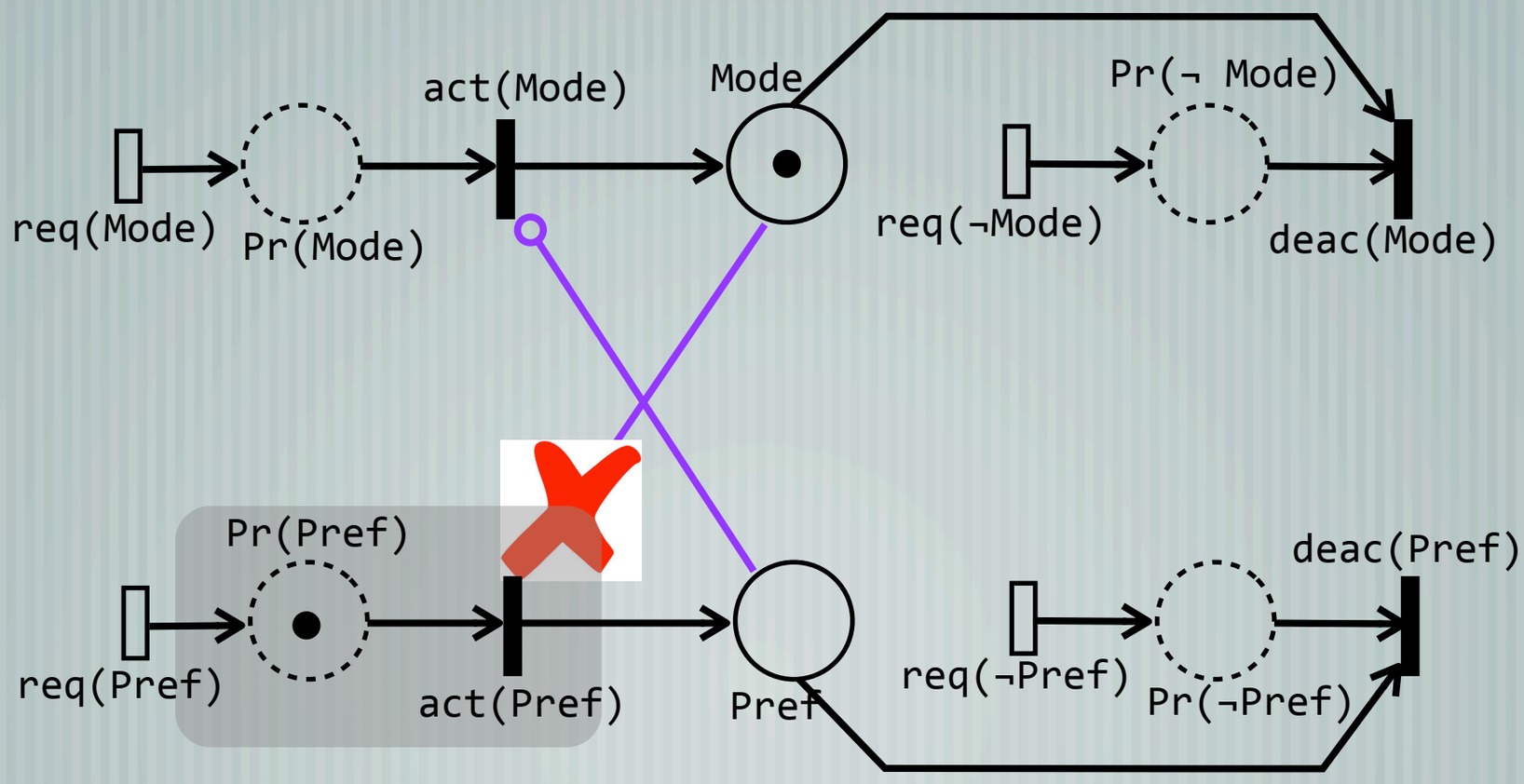


Consistent States

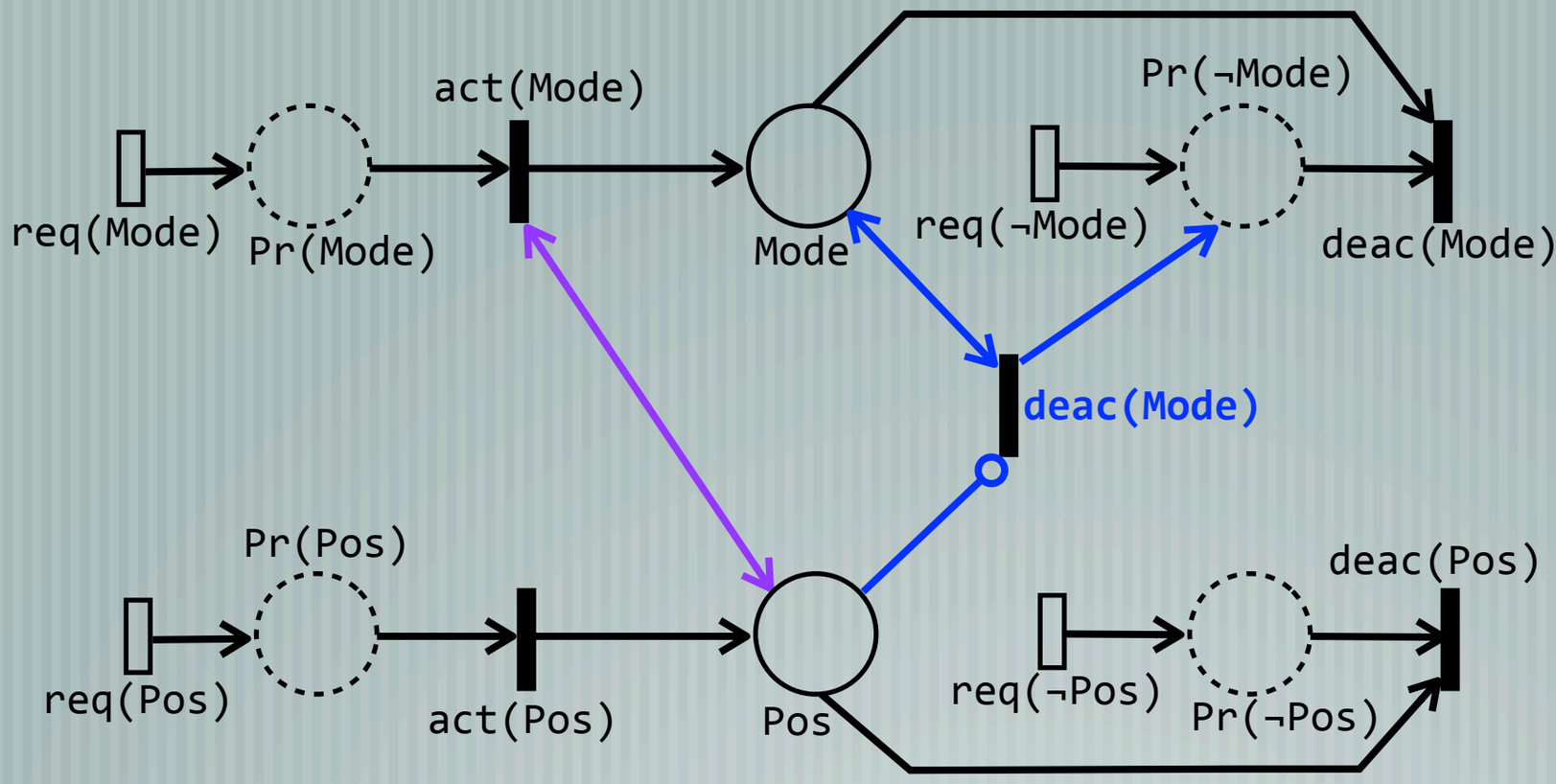
Consistent States

No enabled internal transitions, and no marked temporary places

```
[addExclusionBetween: Mode and: Pref]
```

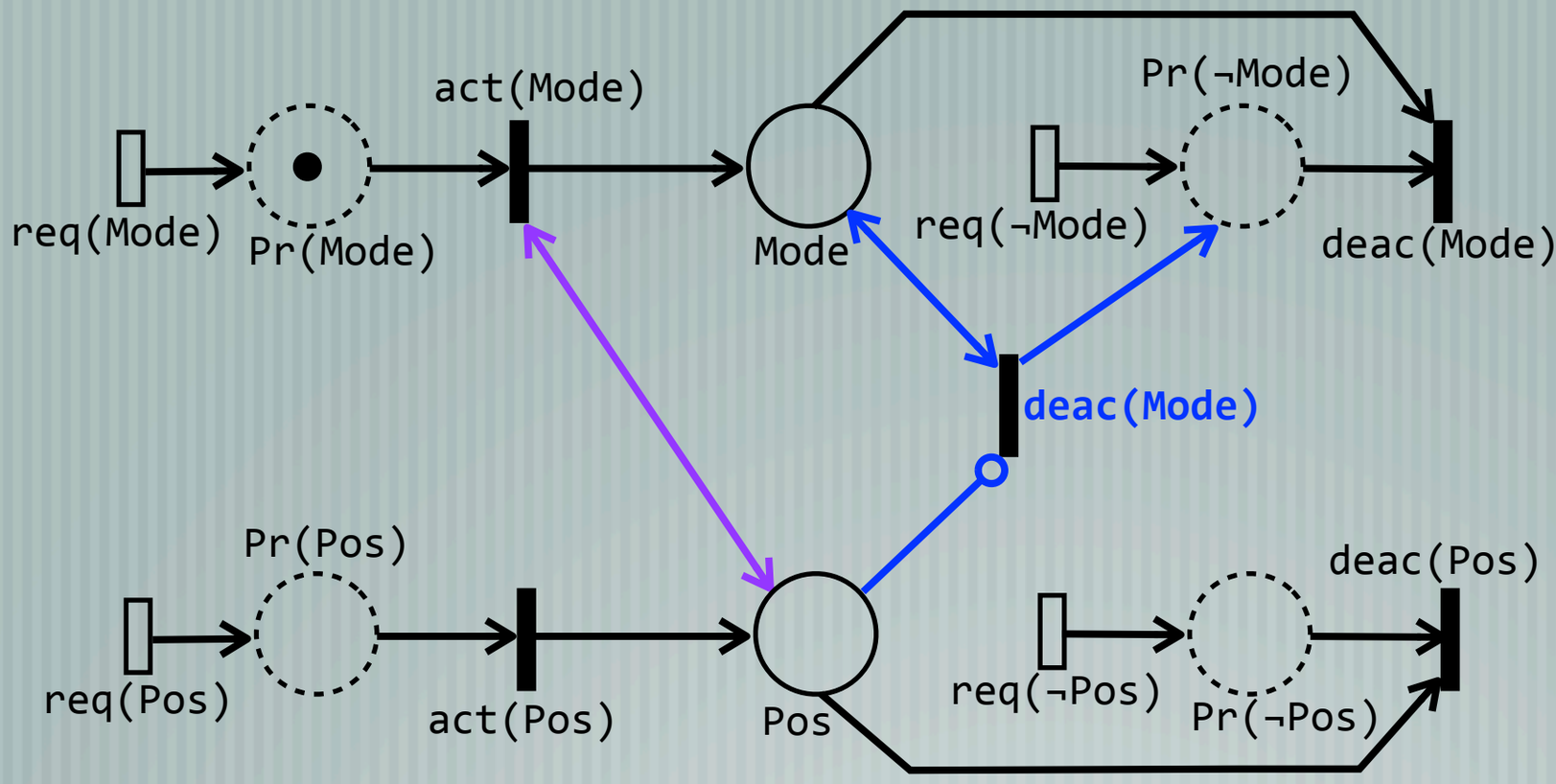


Activation Semantics of CoPNs



Context Petri Nets: Consistent of Context-dependent Behavior.
Cardozo, Nicolás and Vallejos, Jorge and González, Sebastián and Mens, Kim and D'Hondt, Theo. International Workshop on Petri Nets and Software Engineering (PNSE '12).

Activation Semantics of CoPNs

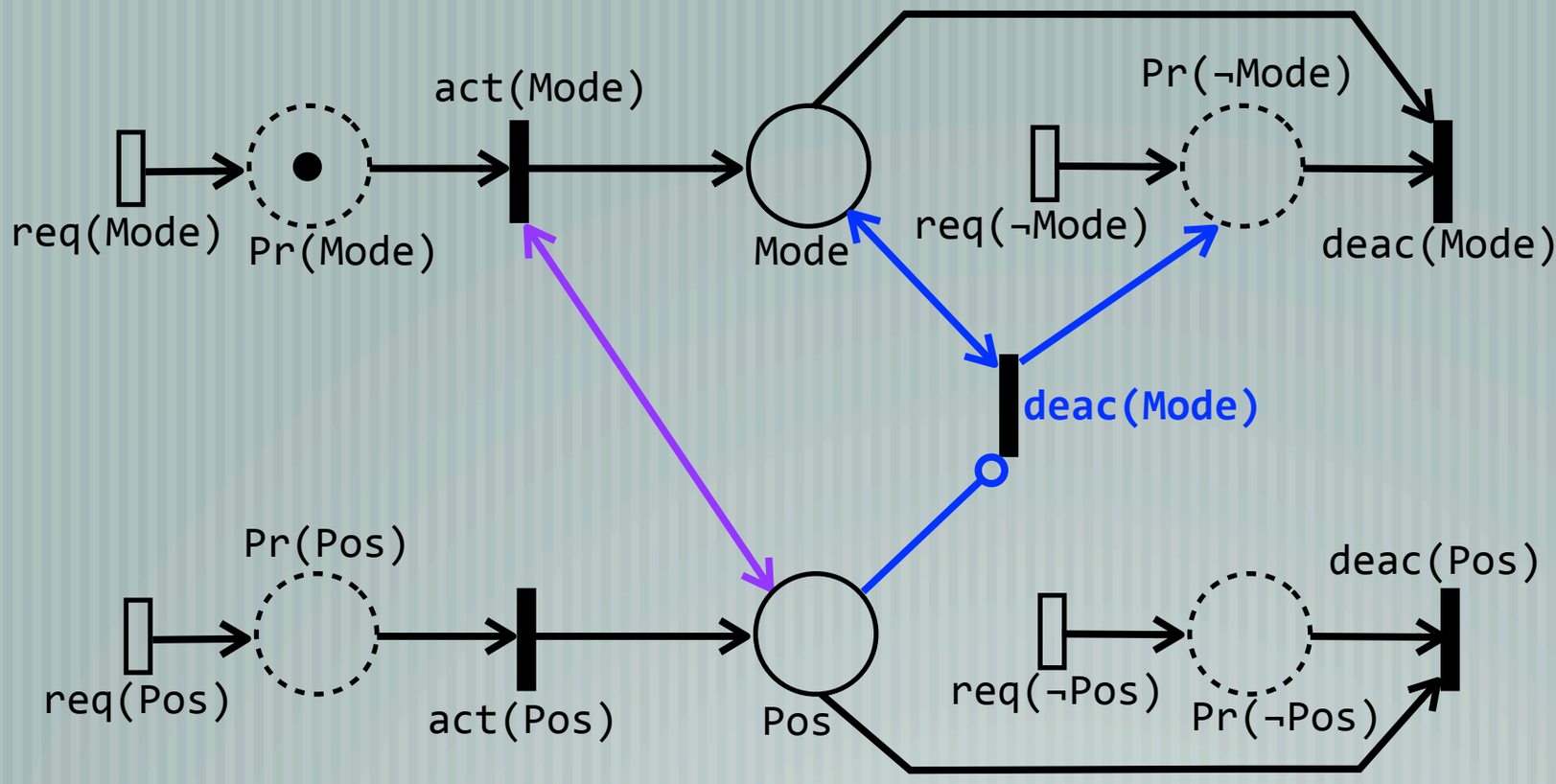


@activate(Mode)

1. External transitions fire after @activate and @deactivate

Context Petri Nets: Consistent of Context-dependent Behavior. *Cardozo, Nicolás and Vallejos, Jorge and González, Sebastián and Mens, Kim and D'Hondt, Theo. International Workshop on Petri Nets and Software Engineering (PNSE '12).*

Activation Semantics of CoPNs

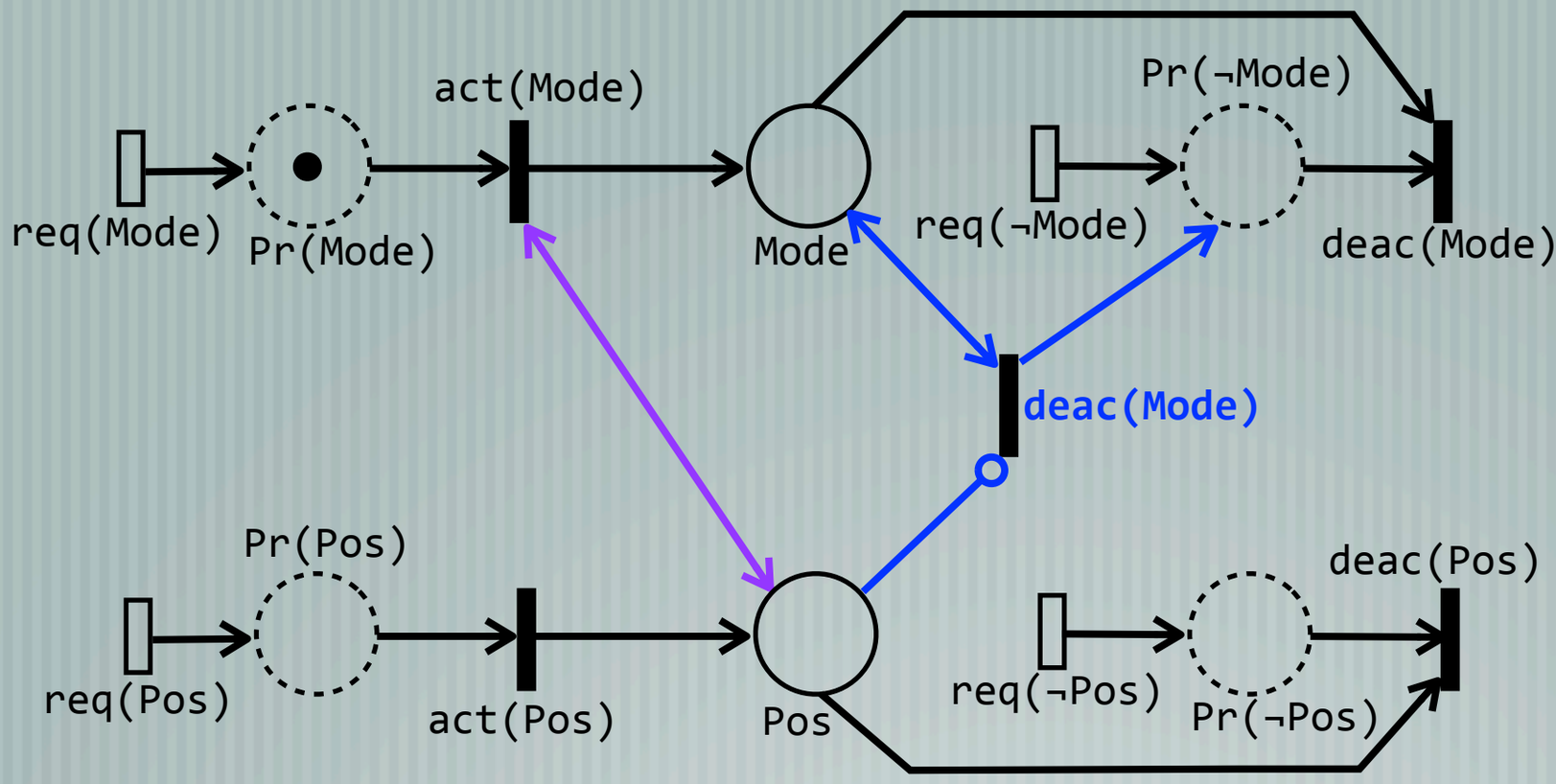


@activate(Mode)

1. External transitions fire after @activate and @deactivate
2. Enabled internal transitions are **always** fired

Context Petri Nets: Consistent of Context-dependent Behavior. *Cardozo, Nicolás and Vallejos, Jorge and González, Sebastián and Mens, Kim and D'Hondt, Theo. International Workshop on Petri Nets and Software Engineering (PNSE '12).*

Activation Semantics of CoPNs

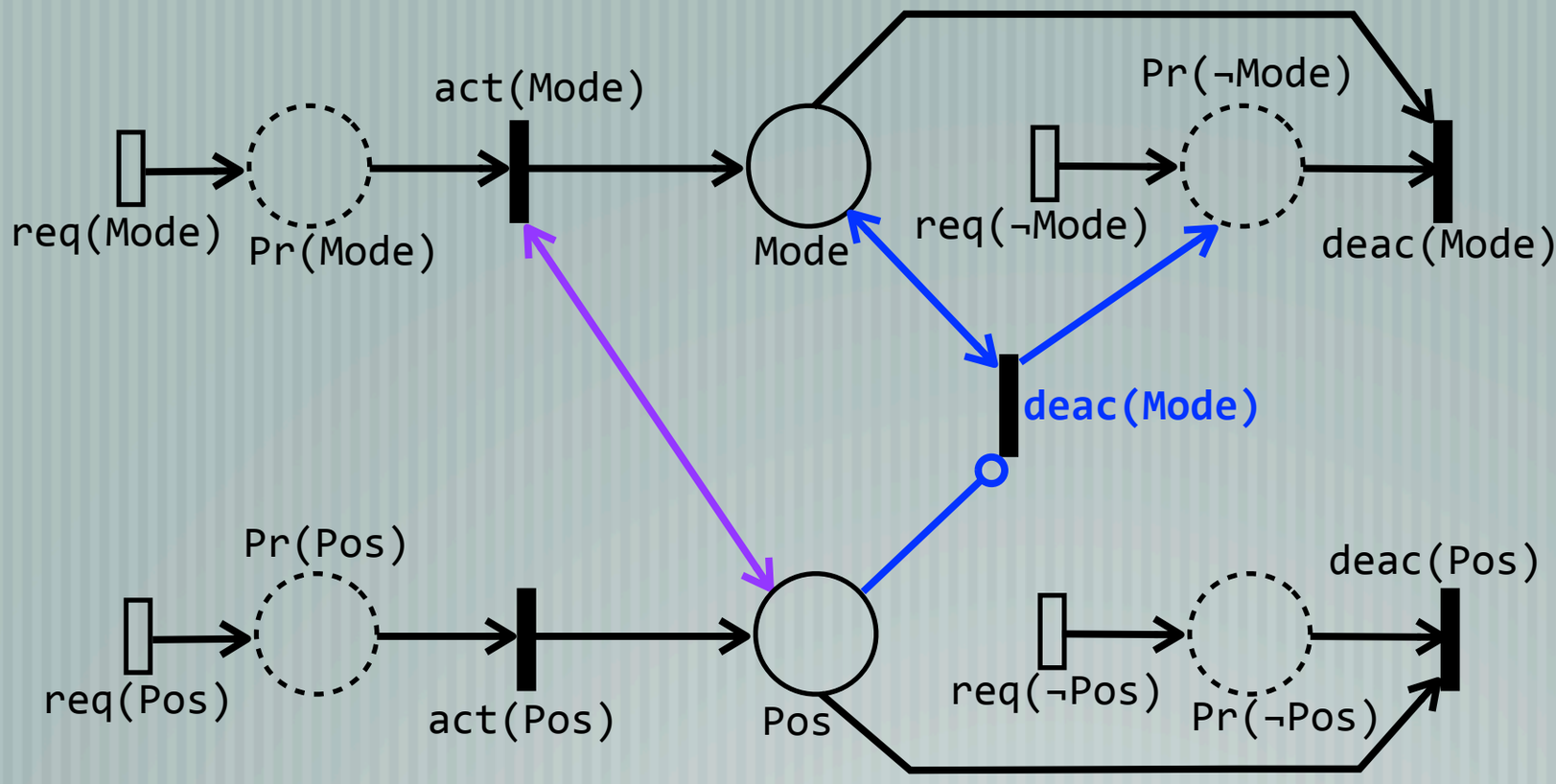


@activate(Mode)

1. External transitions fire after @activate and @deactivate
2. Enabled internal transitions are **always** fired
3. If there is an inconsistency, actions **revert** to previous consistent state

Context Petri Nets: Consistent of Context-dependent Behavior. *Cardozo, Nicolás and Vallejos, Jorge and González, Sebastián and Mens, Kim and D'Hondt, Theo. International Workshop on Petri Nets and Software Engineering (PNSE '12).*

Activation Semantics of CoPNs

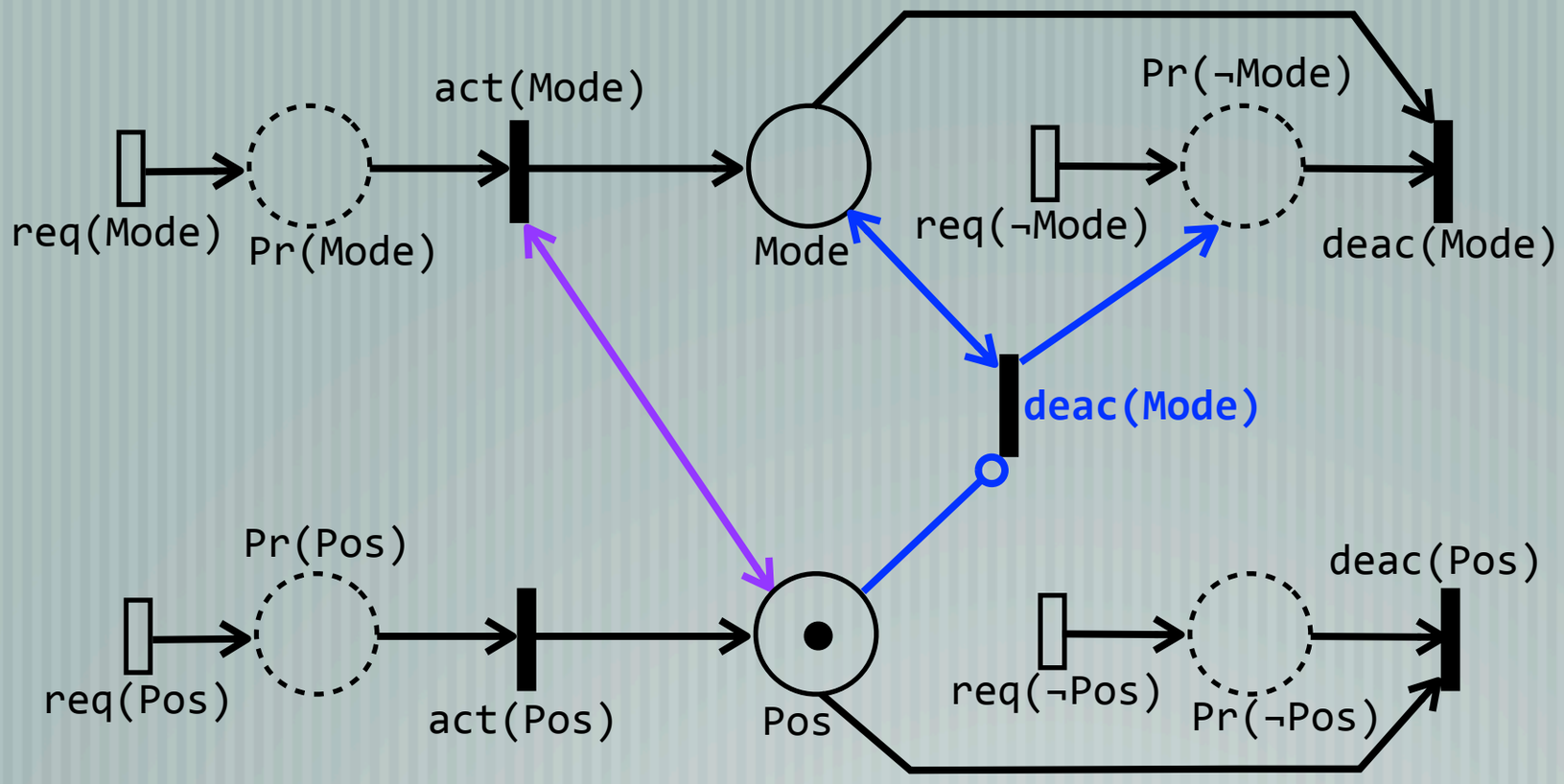


@activate(Mode)

1. External transitions fire after @activate and @deactivate
2. Enabled internal transitions are **always** fired
3. If there is an inconsistency, actions **revert** to previous consistent state

Context Petri Nets: Consistent of Context-dependent Behavior. *Cardozo, Nicolás and Vallejos, Jorge and González, Sebastián and Mens, Kim and D'Hondt, Theo. International Workshop on Petri Nets and Software Engineering (PNSE '12).*

Activation Semantics of CoPNs

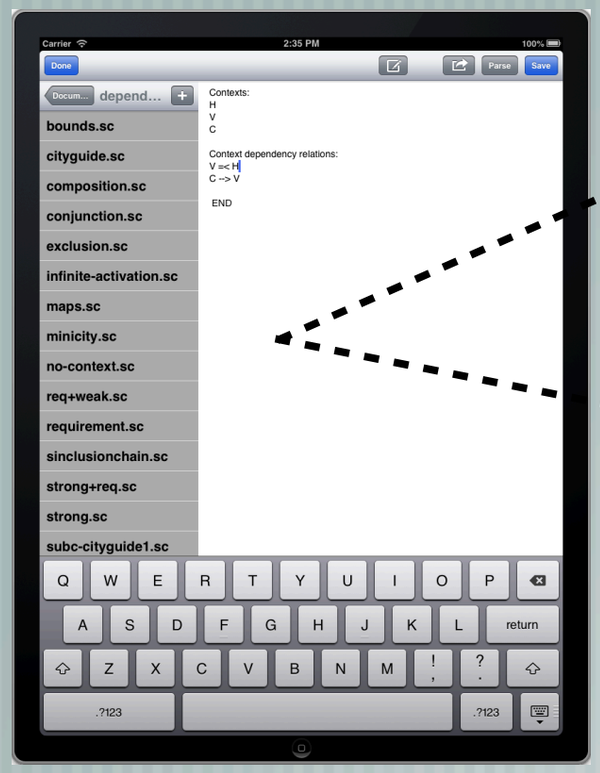



```
@activate(Mode)
@activate(Pos)
```

1. External transitions fire after `@activate` and `@deactivate`
2. Enabled internal transitions are **always** fired
3. If there is an inconsistency, actions **revert** to previous consistent state
4. Consistent states are **accepted**

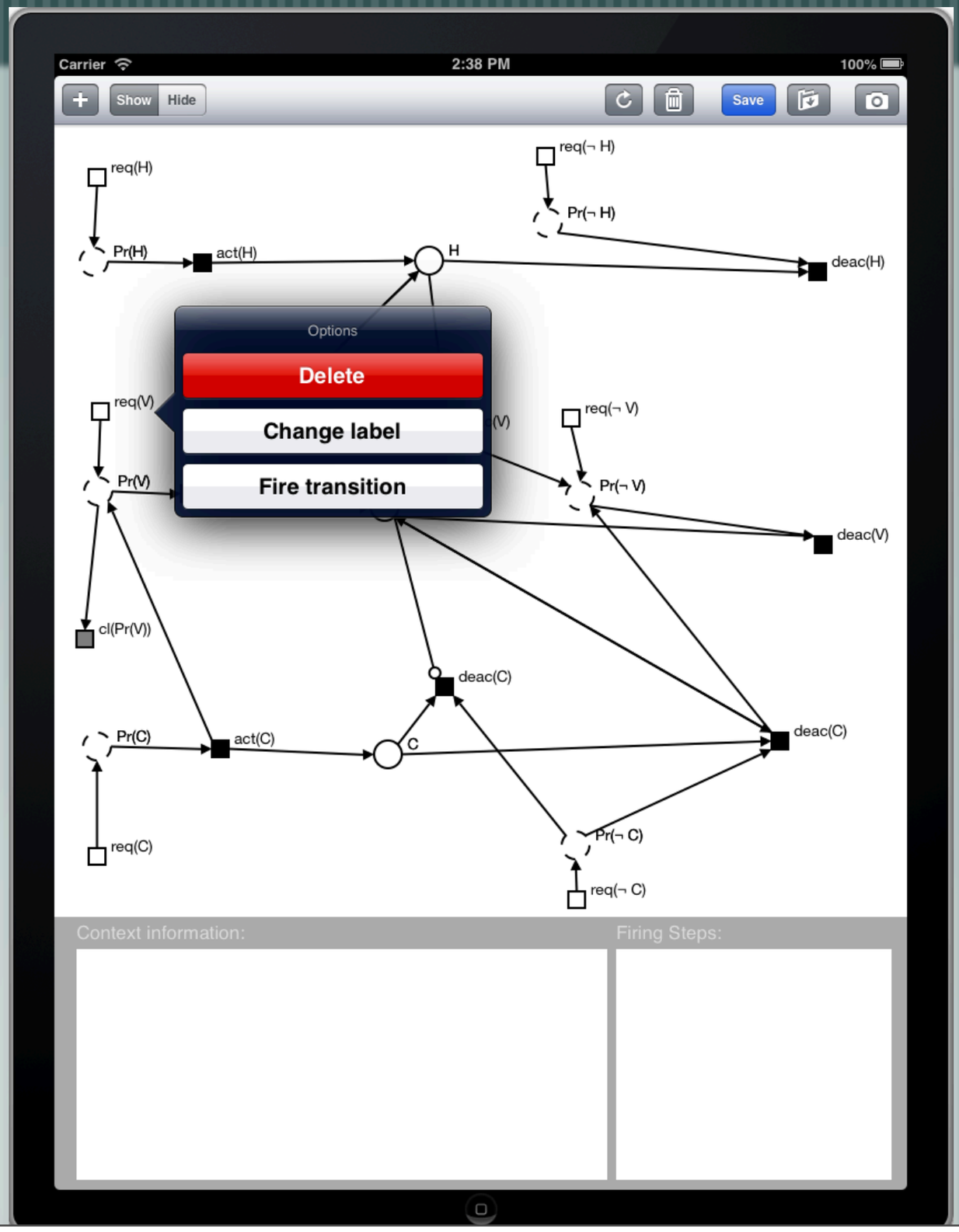
Context Petri Nets: Consistent of Context-dependent Behavior. *Cardozo, Nicolás and Vallejos, Jorge and González, Sebastián and Mens, Kim and D'Hondt, Theo. International Workshop on Petri Nets and Software Engineering (PNSE '12).*

Context Activation Simulation Tool

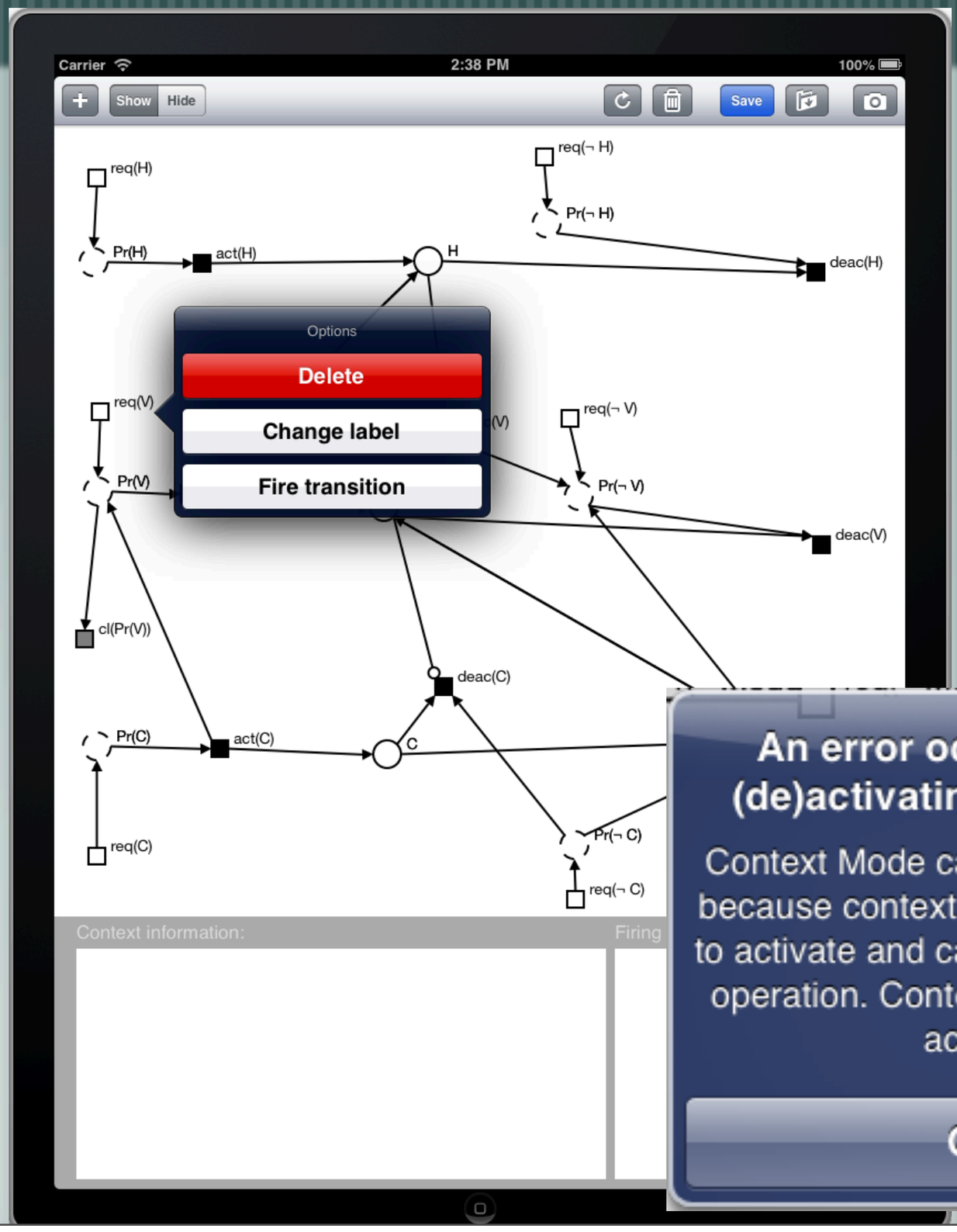



```
Contexts:
Preferences
Mode
FreeWalk
Context dependency relations:
Mode >< Preferences
FreeWalk -> Mode
```

Context Activation Simulation Tool

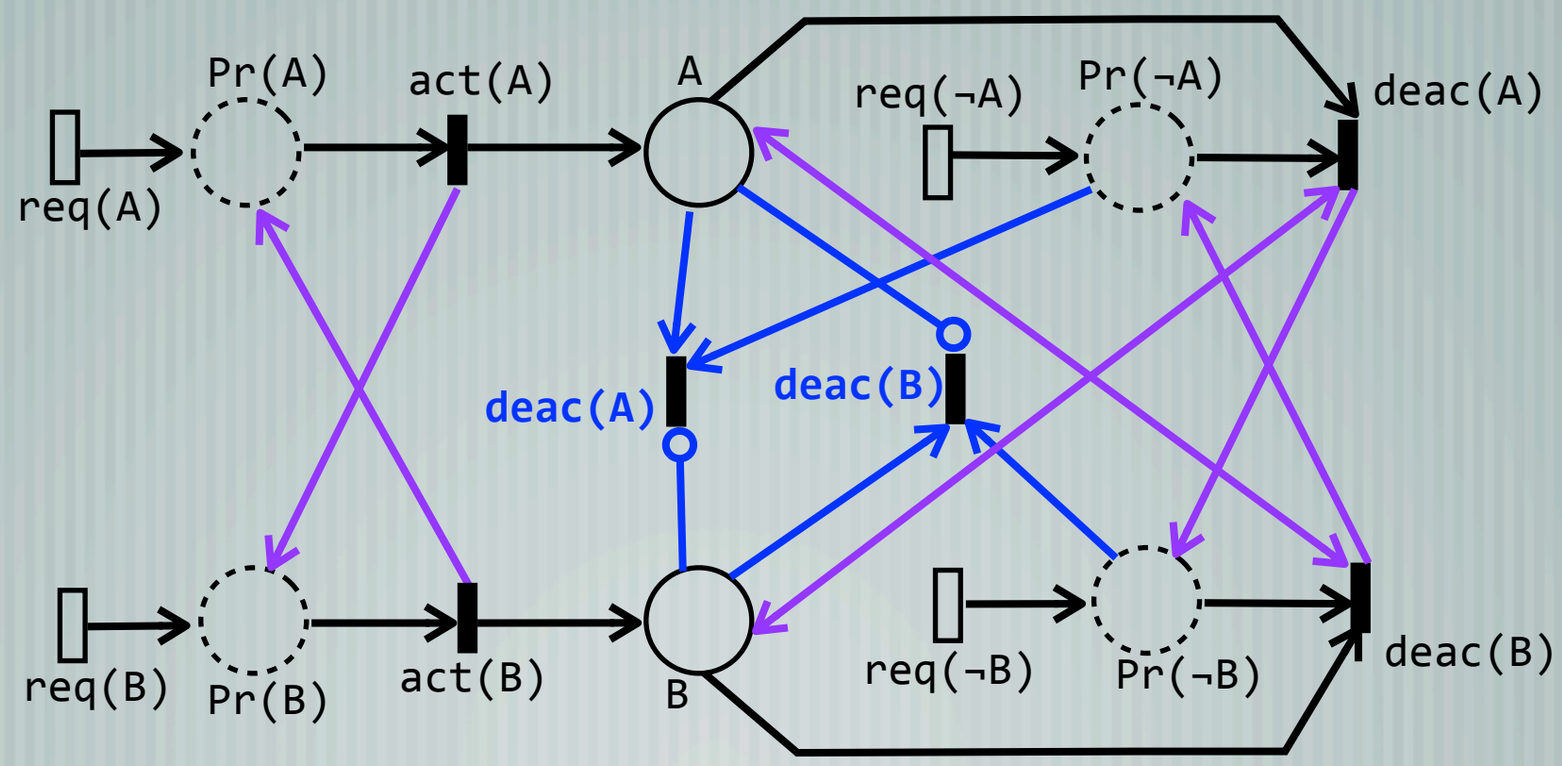


Context Activation Simulation Tool



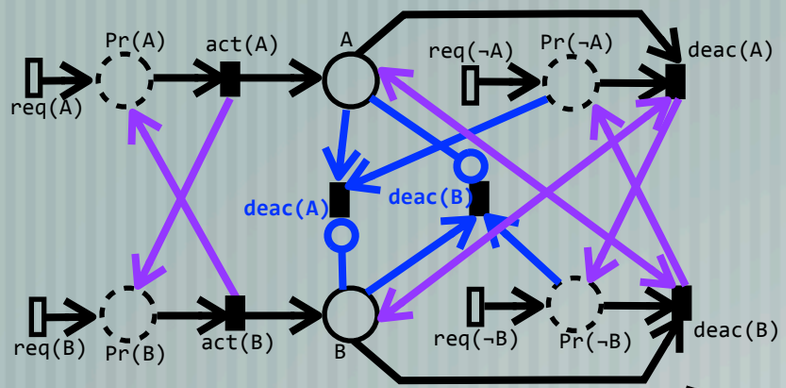
Petri net analyses

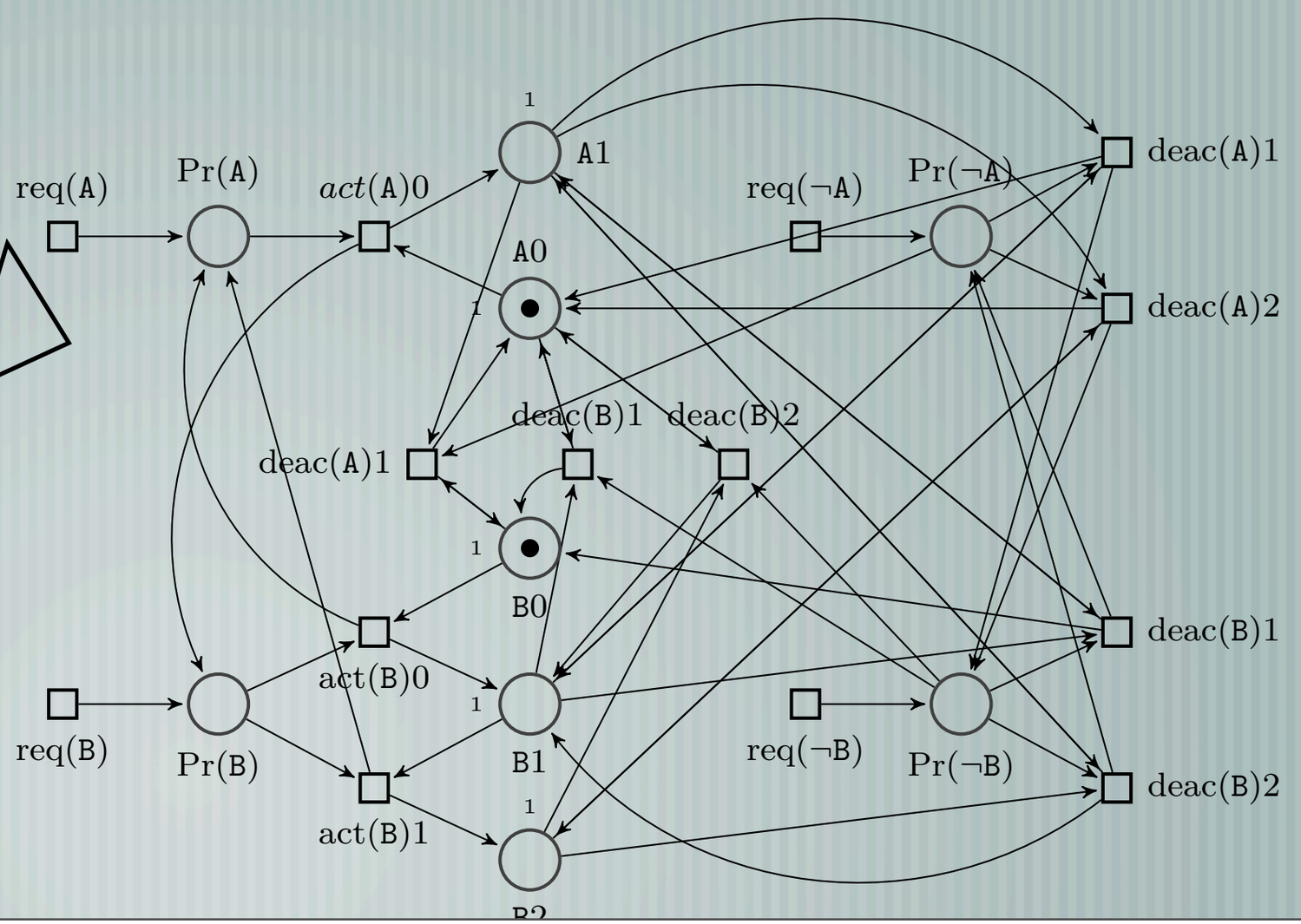
- "Unfold" the CoPN to a traditional Petri net
  - need to introduce bounds => some loss of semantics



Petri net analyses

- "Unfold" the CoPN to a traditional Petri net
  - need to introduce bounds => some loss of semantics

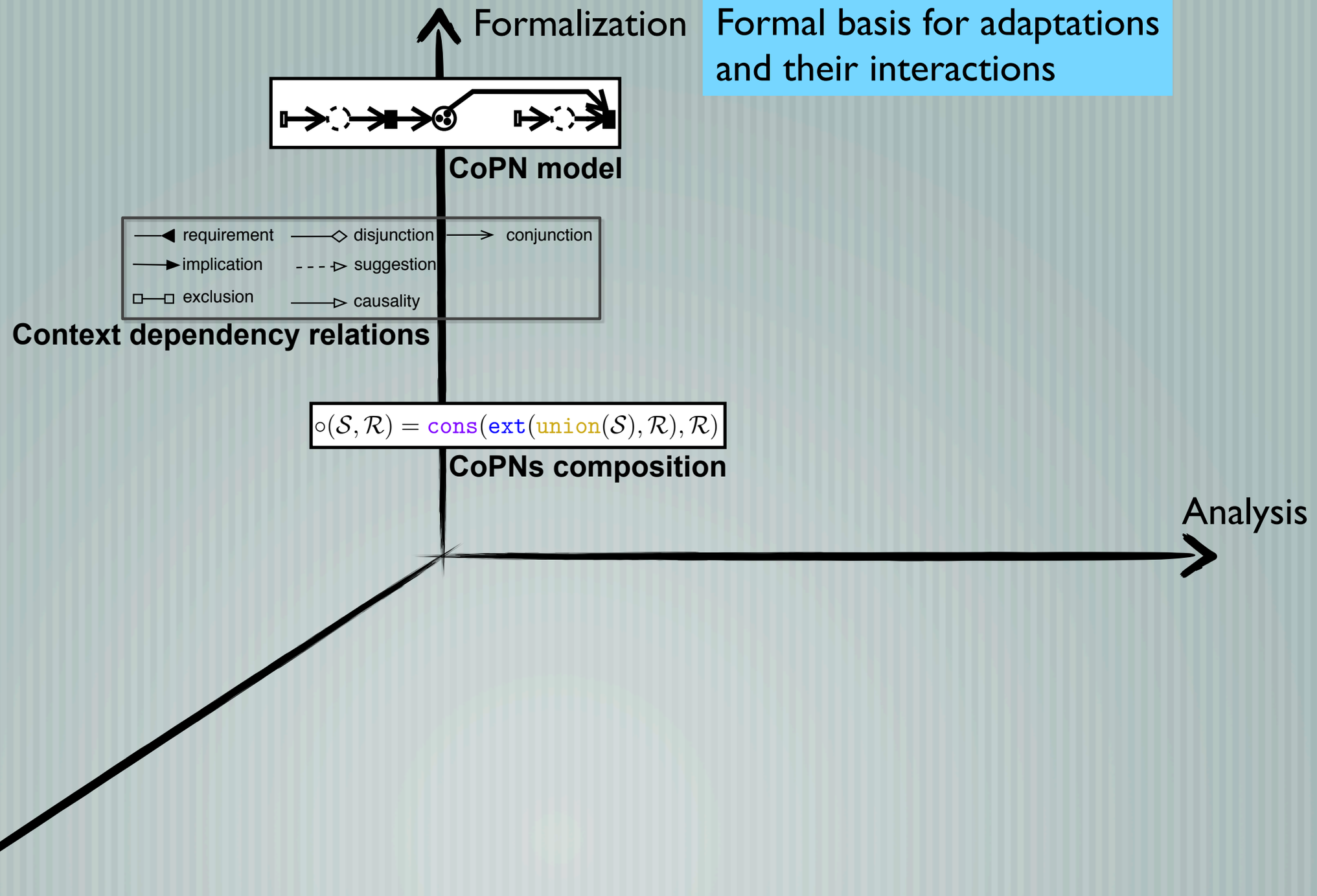




Petri net analyses

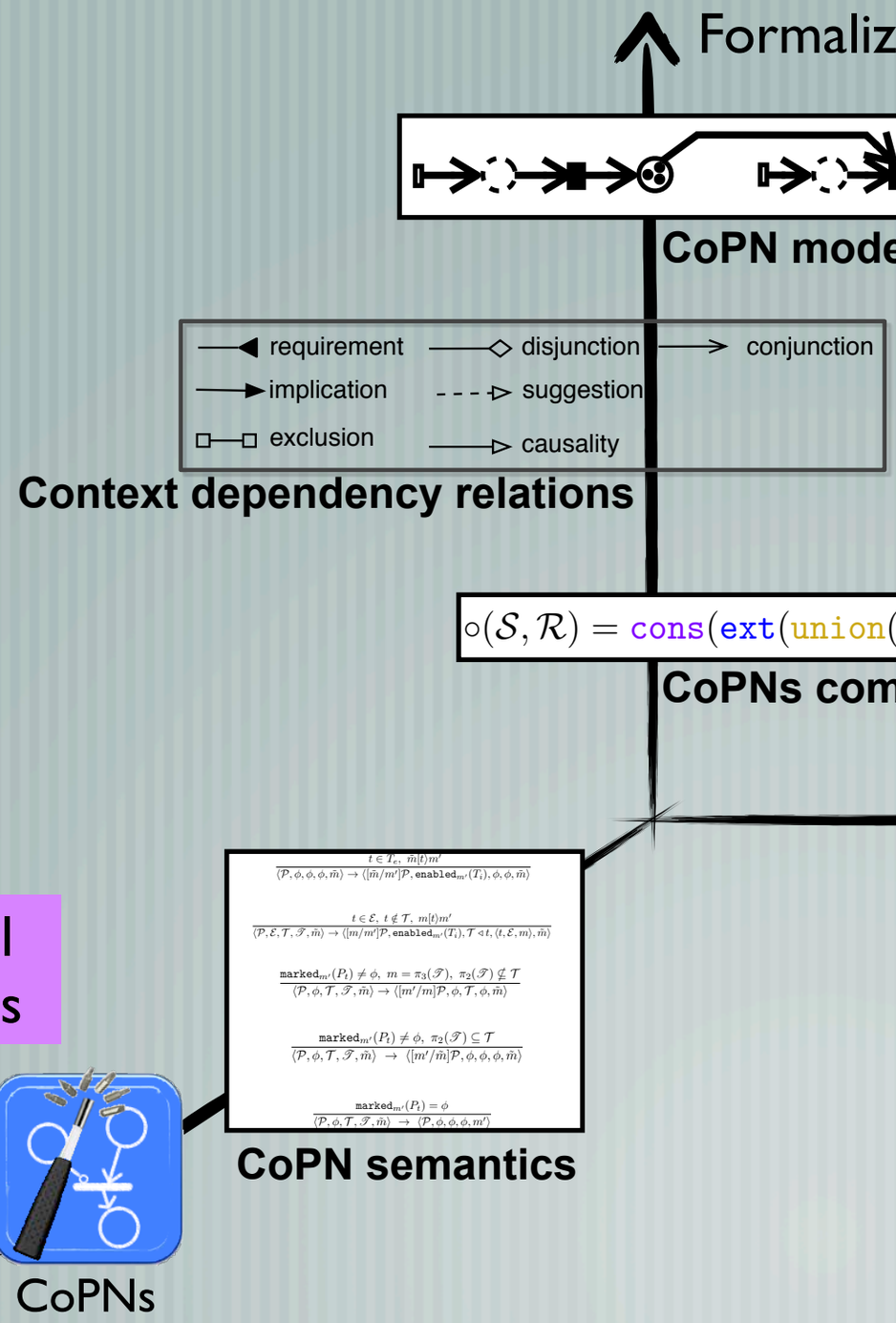
- "Unfold" the CoPN to a traditional Petri net
- Use a traditional Petri net analysis tool like LoLA
  -  Choose appropriate bounds to avoid too large Petri nets
- Use LoLa to reason about properties like reachability and liveness
  - partly manual (guided by the user)
  -  partly automated (analysis tests generated from dependencies)

Conclusion



Formal basis for adaptations and their interactions

Conclusion



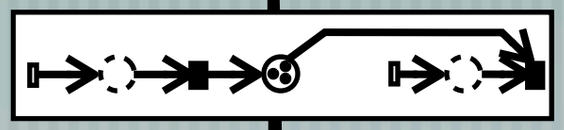
Formal basis for adaptations and their interactions

Execution model for COP systems

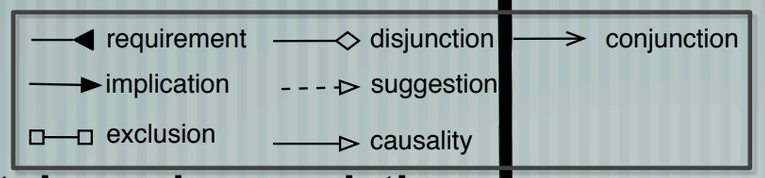
Conclusion

Formalization

Formal basis for adaptations and their interactions



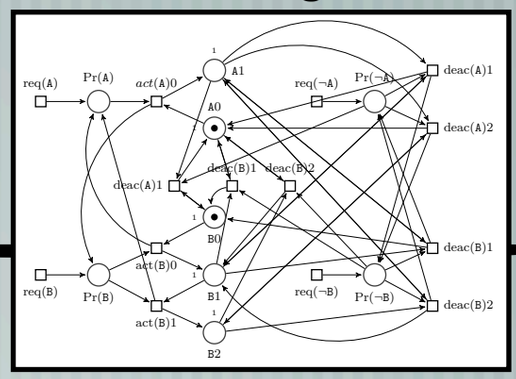
CoPN model



Context dependency relations

Reasoning and analysis over dynamic adaptations

Reasoning module

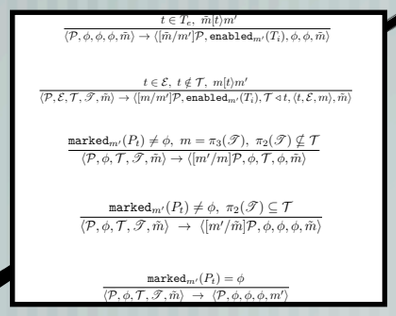


Analysis

$$\circ(\mathcal{S},\mathcal{R}) = \texttt{cons}(\texttt{ext}(\texttt{union}(\mathcal{S}),\mathcal{R}),\mathcal{R})$$

CoPNs composition

Execution model for COP systems



$$\frac{t \in \mathcal{T}_e,\ \bar{m}[t\rangle m'}{\langle \mathcal{P},\phi,\phi,\phi,\bar{m}\rangle \rightarrow \langle [\bar{m}/m']\mathcal{P}, \texttt{enabled}_{m'}(T_t),\phi,\phi,\bar{m}\rangle}$$

$$\frac{t \in \mathcal{E},\ t \notin \mathcal{T},\ m[t\rangle m'}{\langle \mathcal{P},\mathcal{E},\mathcal{T},\mathcal{F},\bar{m}\rangle \rightarrow \langle [m/m']\mathcal{P}, \texttt{enabled}_{m'}(T_t), \mathcal{T} \lhd t, \langle t,\mathcal{E},m\rangle,\bar{m}\rangle}$$

$$\frac{\texttt{marked}_{m'}(P_t) \neq \phi,\ m = \pi_3(\mathcal{F}),\ \pi_2(\mathcal{F}) \nsubseteq \mathcal{T}}{\langle \mathcal{P},\phi,\mathcal{T},\mathcal{F},\bar{m}\rangle \rightarrow \langle [m'/m]\mathcal{P},\phi,\mathcal{T},\phi,\bar{m}\rangle}$$

$$\frac{\texttt{marked}_{m'}(P_t) \neq \phi,\ \pi_2(\mathcal{F}) \subseteq \mathcal{T}}{\langle \mathcal{P},\phi,\mathcal{T},\mathcal{F},\bar{m}\rangle \ \rightarrow\ \langle [m'/\bar{m}]\mathcal{P},\phi,\phi,\phi,\bar{m}\rangle}$$

$$\frac{\texttt{marked}_{m'}(P_t) = \phi}{\langle \mathcal{P},\phi,\mathcal{T},\mathcal{F},\bar{m}\rangle \ \rightarrow\ \langle \mathcal{P},\phi,\phi,\phi,m'\rangle}$$

CoPN semantics

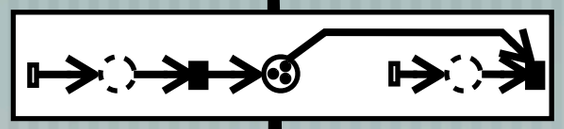
Execution

CoPNs

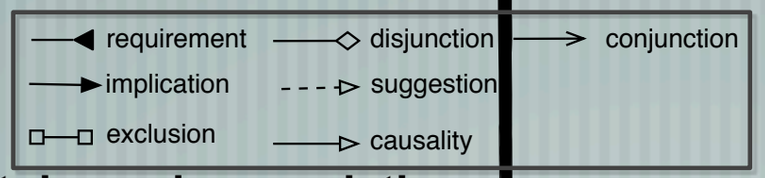
Conclusion

Formalization

Formal basis for adaptations and their interactions



CoPN model



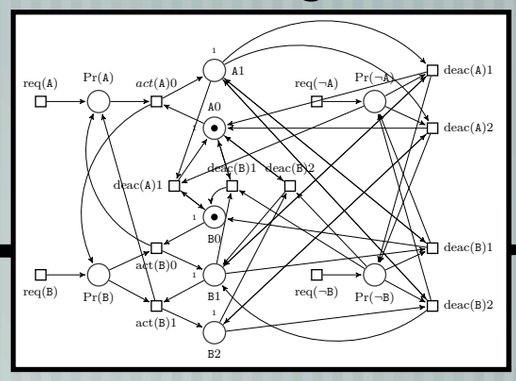
Context dependency relations

$$\circ(\mathcal{S},\mathcal{R}) = \texttt{cons}(\texttt{ext}(\texttt{union}(\mathcal{S}),\mathcal{R}),\mathcal{R})$$

CoPNs composition

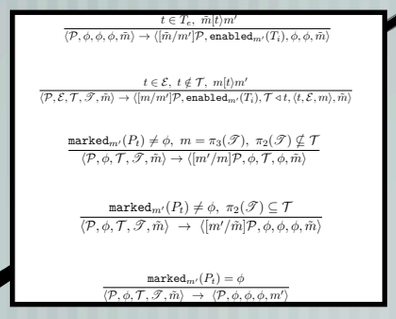
Reasoning and analysis over dynamic adaptations

Reasoning module



Analysis

Execution model for COP systems



$$\frac{t \in \mathcal{T}_e,\ \bar{m}[t\rangle m'}{\langle \mathcal{P},\phi,\phi,\phi,\bar{m}\rangle \rightarrow \langle [\bar{m}/m']\mathcal{P}, \texttt{enabled}_{m'}(T_i),\phi,\phi,\bar{m}\rangle}$$

$$\frac{t \in \mathcal{E},\ t \notin \mathcal{T},\ m[t\rangle m'}{\langle \mathcal{P},\mathcal{E},\mathcal{T},\mathcal{F},\bar{m}\rangle \rightarrow \langle [m/m']\mathcal{P}, \texttt{enabled}_{m'}(T_i), \mathcal{T} \lhd t, \langle t,\mathcal{E},m\rangle,\bar{m}\rangle}$$

$$\frac{\texttt{marked}_{m'}(P_t) \neq \phi,\ m = \pi_3(\mathcal{F}),\ \pi_2(\mathcal{F}) \nsubseteq \mathcal{T}}{\langle \mathcal{P},\phi,\mathcal{T},\mathcal{F},\bar{m}\rangle \rightarrow \langle [m'/m]\mathcal{P},\phi,\mathcal{T},\phi,\bar{m}\rangle}$$

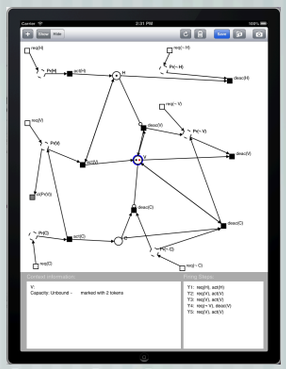
$$\frac{\texttt{marked}_{m'}(P_t) \neq \phi,\ \pi_2(\mathcal{F}) \subseteq \mathcal{T}}{\langle \mathcal{P},\phi,\mathcal{T},\mathcal{F},\bar{m}\rangle \ \rightarrow\ \langle [m'/\bar{m}]\mathcal{P},\phi,\phi,\phi,\bar{m}\rangle}$$

$$\frac{\texttt{marked}_{m'}(P_t) = \phi}{\langle \mathcal{P},\phi,\mathcal{T},\mathcal{F},\bar{m}\rangle \ \rightarrow\ \langle \mathcal{P},\phi,\phi,\phi,m'\rangle}$$

CoPN semantics

Execution

CoPNs



Simulation tool

Simulation of interactions between contexts


RELEASED