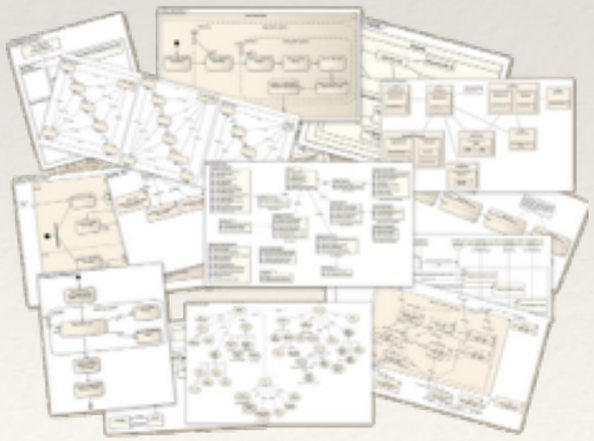
Verifying Incomplete and Evolving Specifications

Claudio Menghi
menghi@elet.polimi.it

Verifying Specifications

Requirements ϕ

Specification M



Verification

yes

no +

counterexample

Evolving

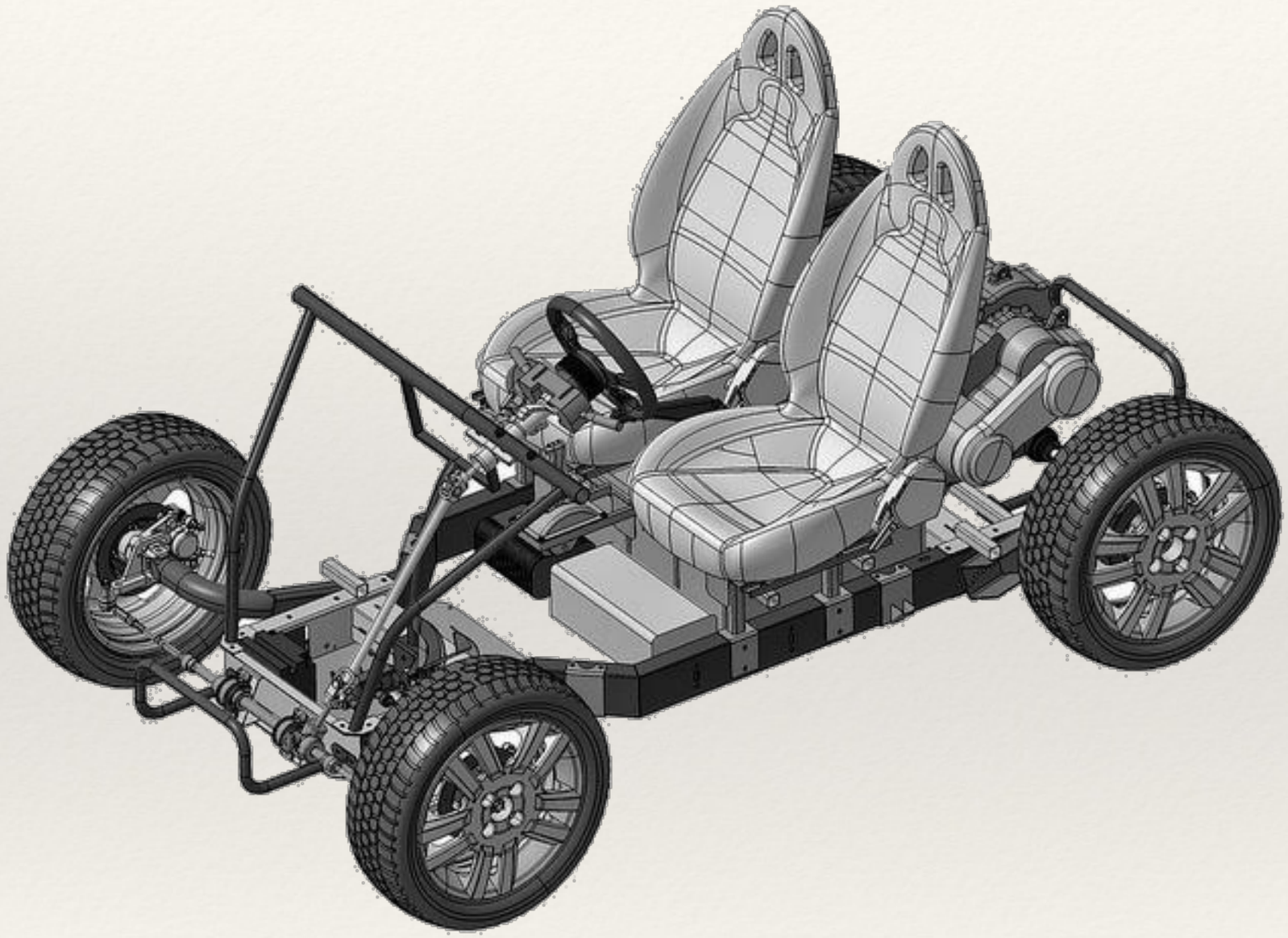

Quality
Price
Success
Sale
Production
Development
Bussiness Plan
Analysis
Bussiness Idea
IDEA
BUSINESS PLAN
PRODUCTION
SUCCESS
NO
YES
Cloud Serv
main ()
{
printf ("Business Idea");
}
?
IF WORK
NO
YES
Profit
Market Analysis

Evolving

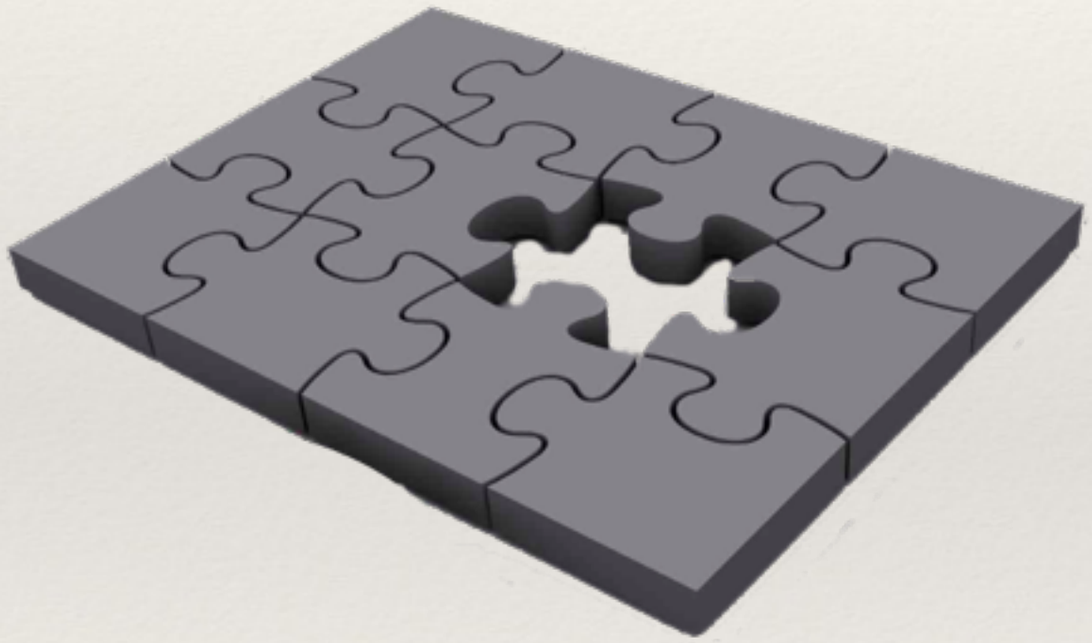

9:42 AM
New
What's Hot
facebook
Jetset Expenses
PocketGuitar
Tipulator - The Fun...
Rocket Taxi call a...


Quality
Price
Success
Sale
Production
Development
Bussiness
Business Plan
Analysis
Bussiness Idea
main ()
{
printf ("Business Idea");
}
IDEA
BUSINESS PLAN
PRODUCTION
SUCCESS
YES
NO
IF WORK
Profit
Market Analysis

Incomplete



Incomplete and Evolving



Incompleteness

Evolution

Verifying Incomplete and Evolving Specifications

Requirements

Specification

Verification

yes

no + count

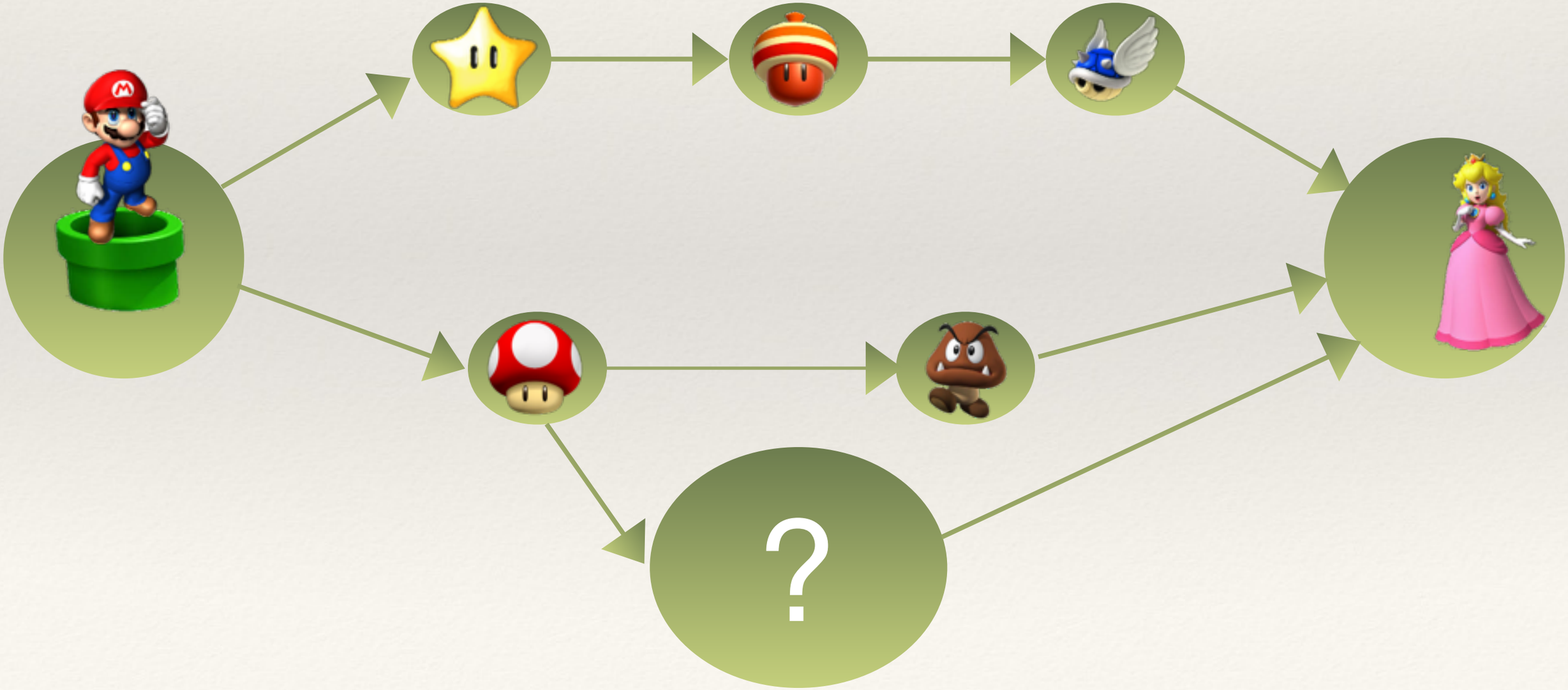
maybe + constraints

Constraints

Constraints are conditions (necessary and sufficient) that, if satisfied, guarantee that ϕ holds for M.

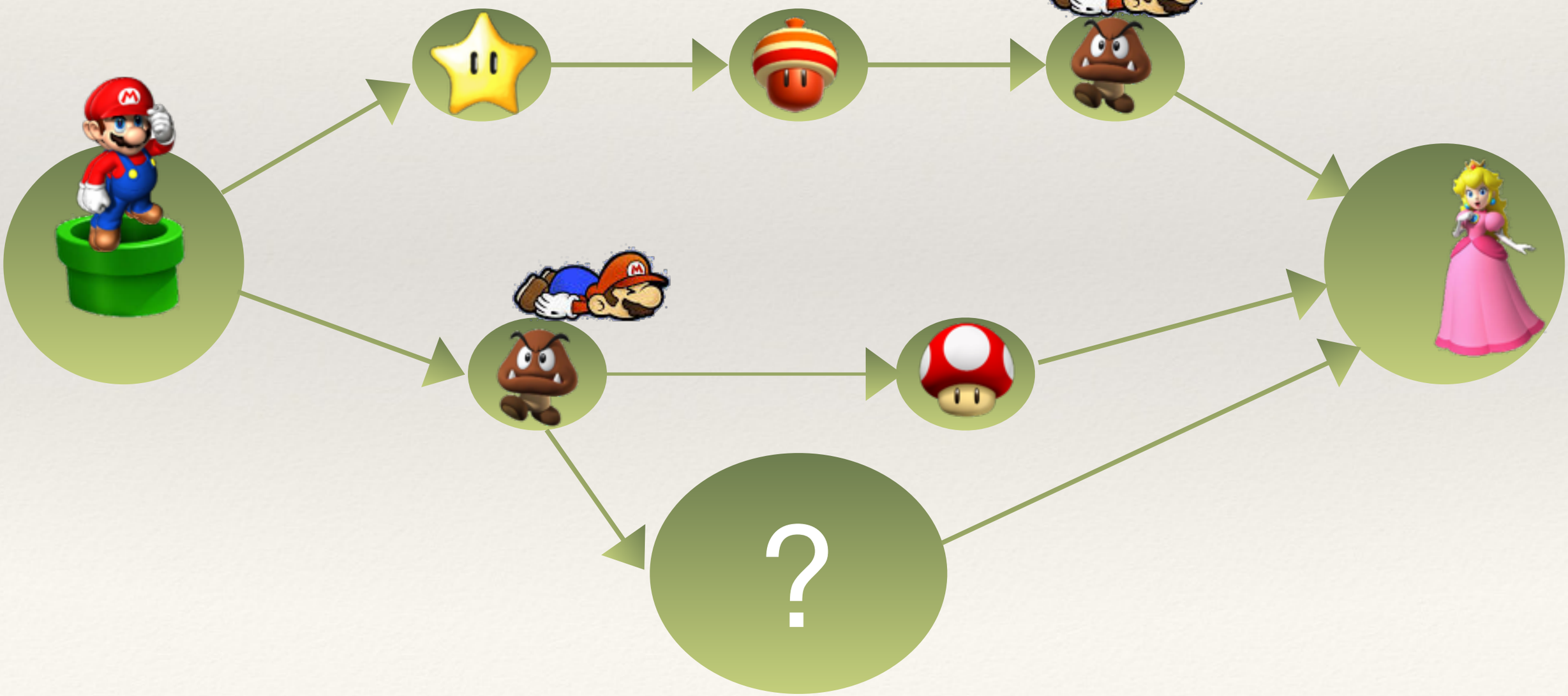
The idea in practice

E(¬())U()



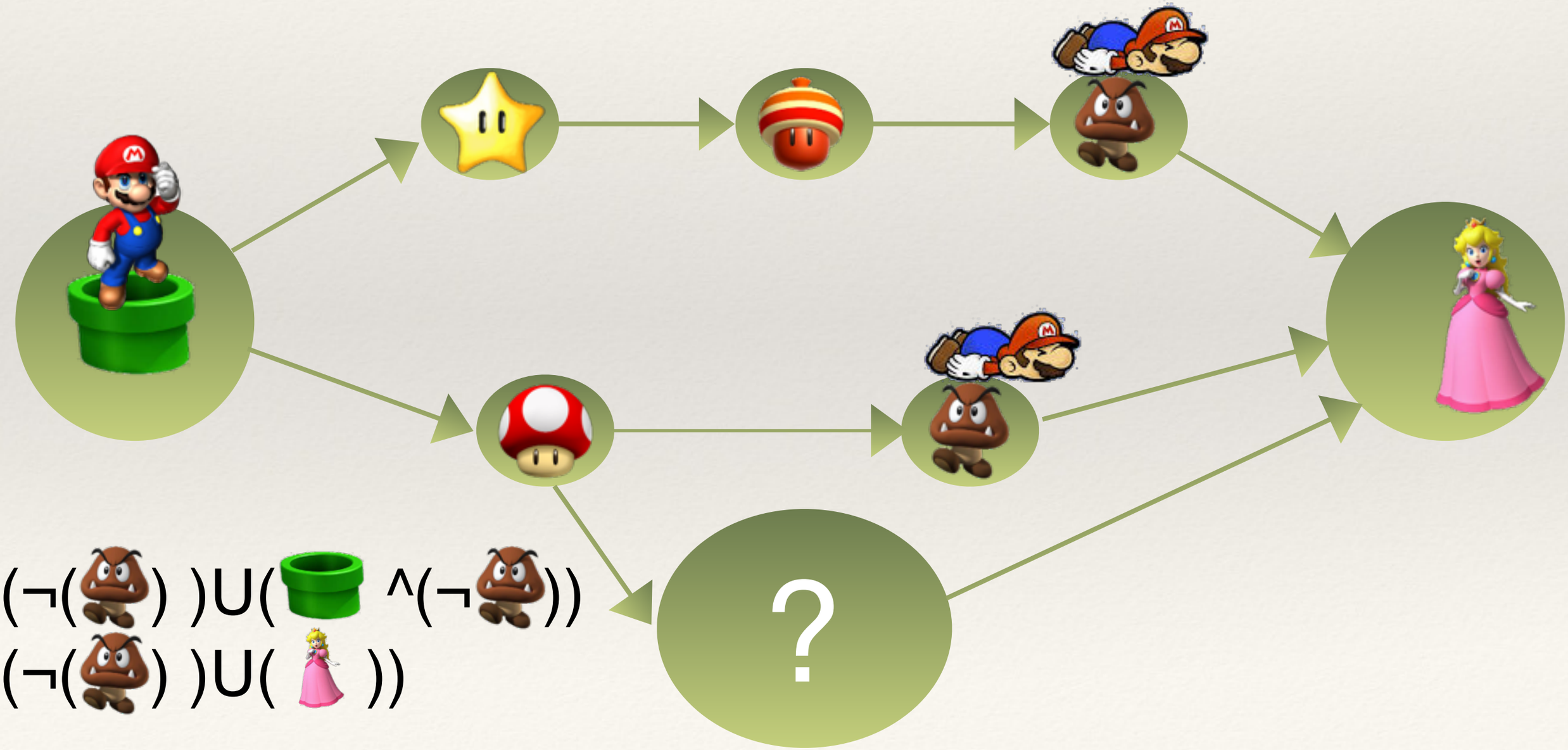
The idea in practice

E(¬())U()



The idea in practice

E(¬())U()

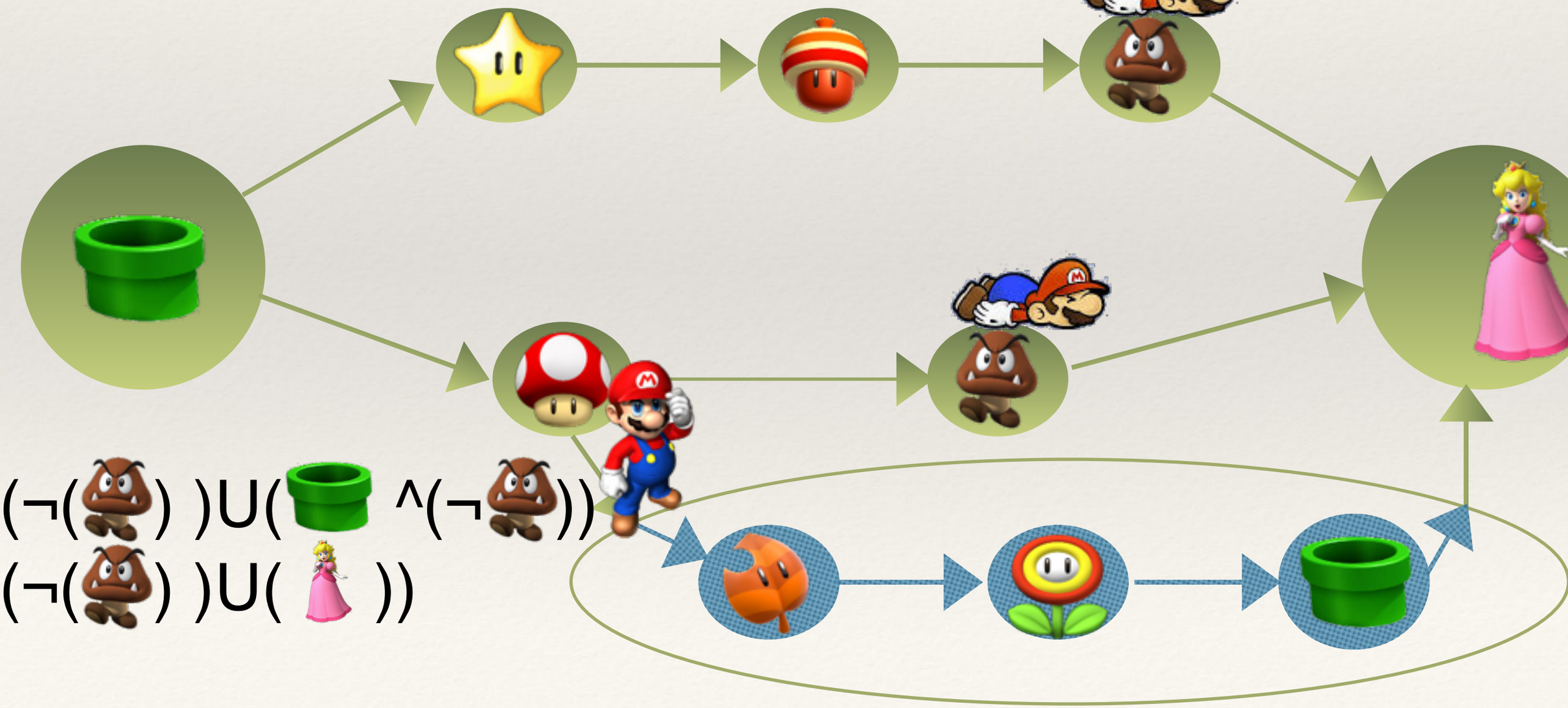


E(¬( ))U(^(¬))

E(¬( ))U( ))

The idea in practice

E(¬())U()

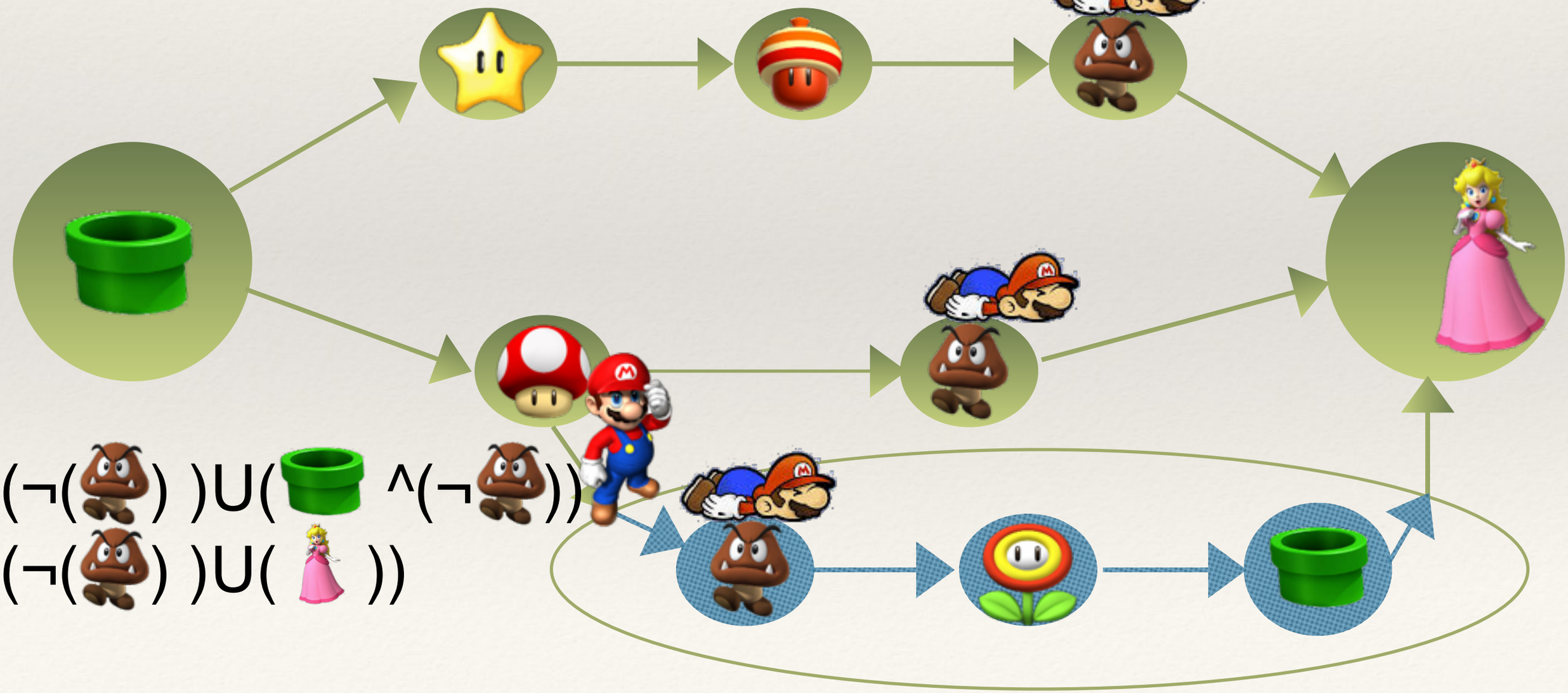


E(¬( ))U(^(¬))

E(¬( ))U( ))

The idea in practice

E(¬()U()



E(¬( )U(^(¬))

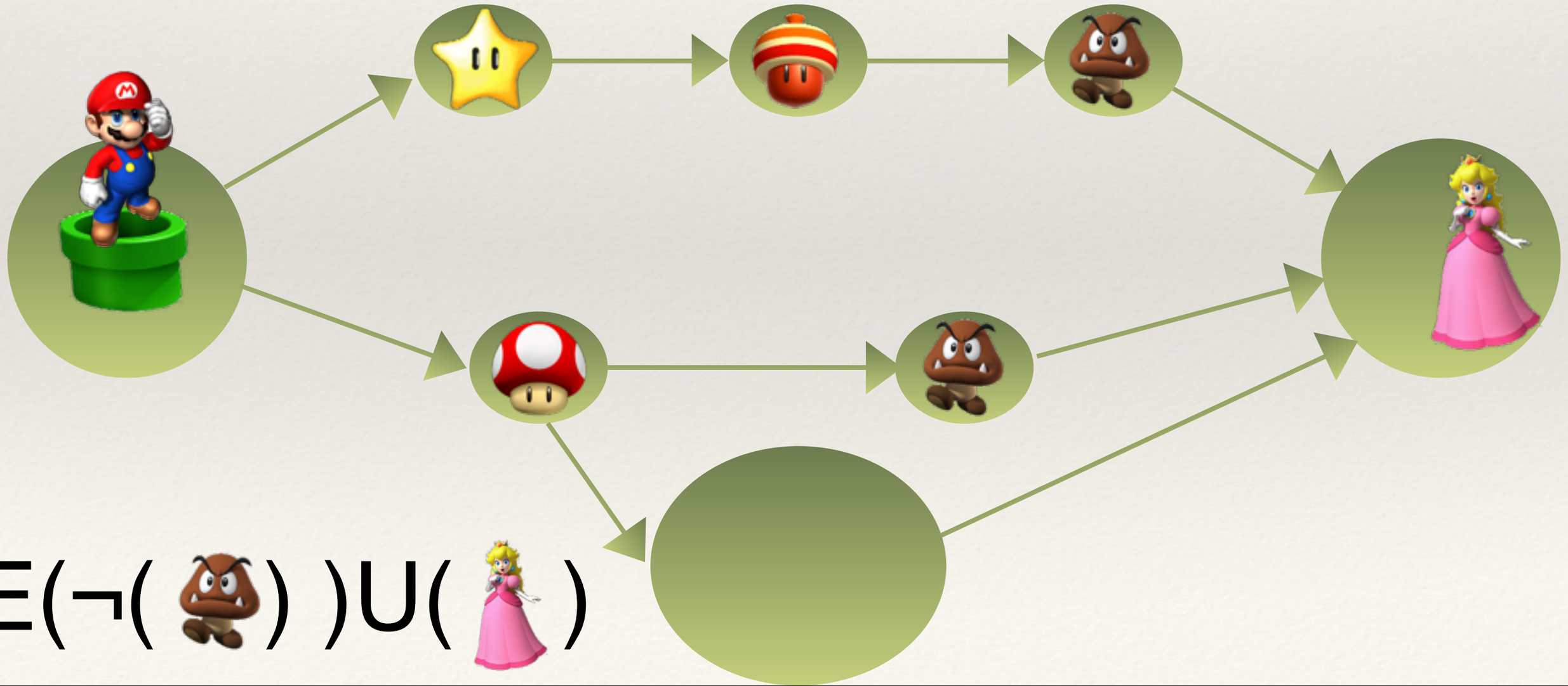
E(¬( )U( ))

Consideration of our Approach

The **efficiency** of the approach
strongly depends
on the number and on the type
of the **constraints** generated

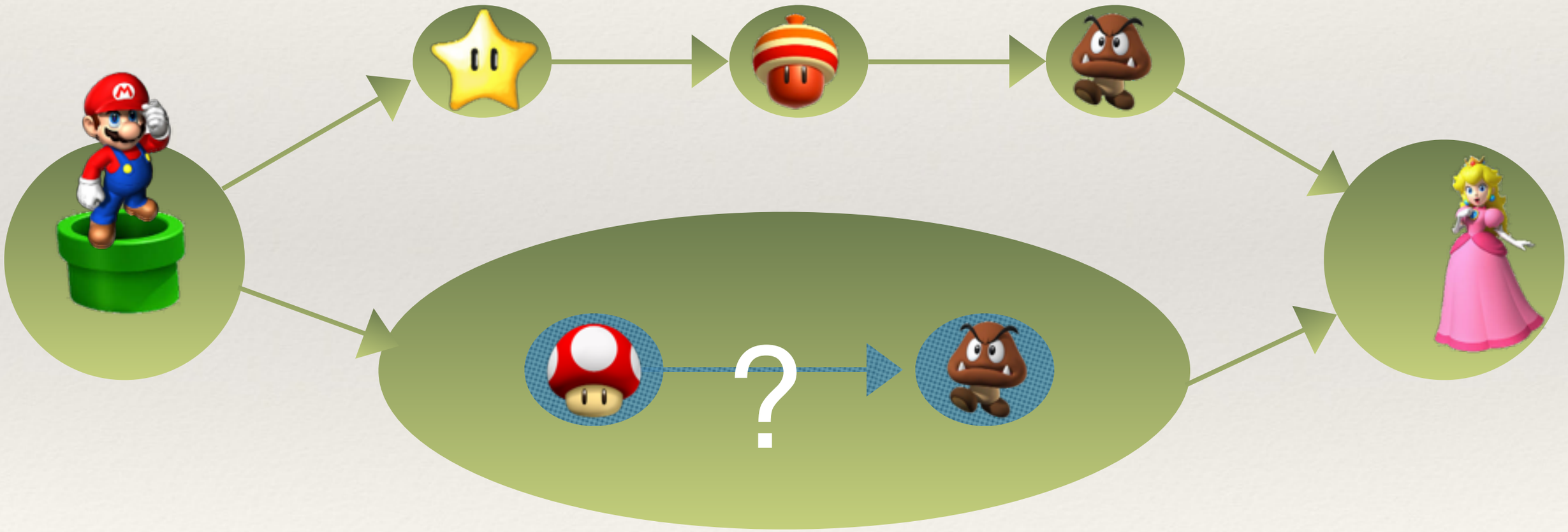
Consideration of our Approach

(1) it is possible to check whether an ***initial, incomplete and high level*** *description of the system (such as the one produced during earlier stages of the software design)* ***satisfies a given property****.*



Consideration of our Approach

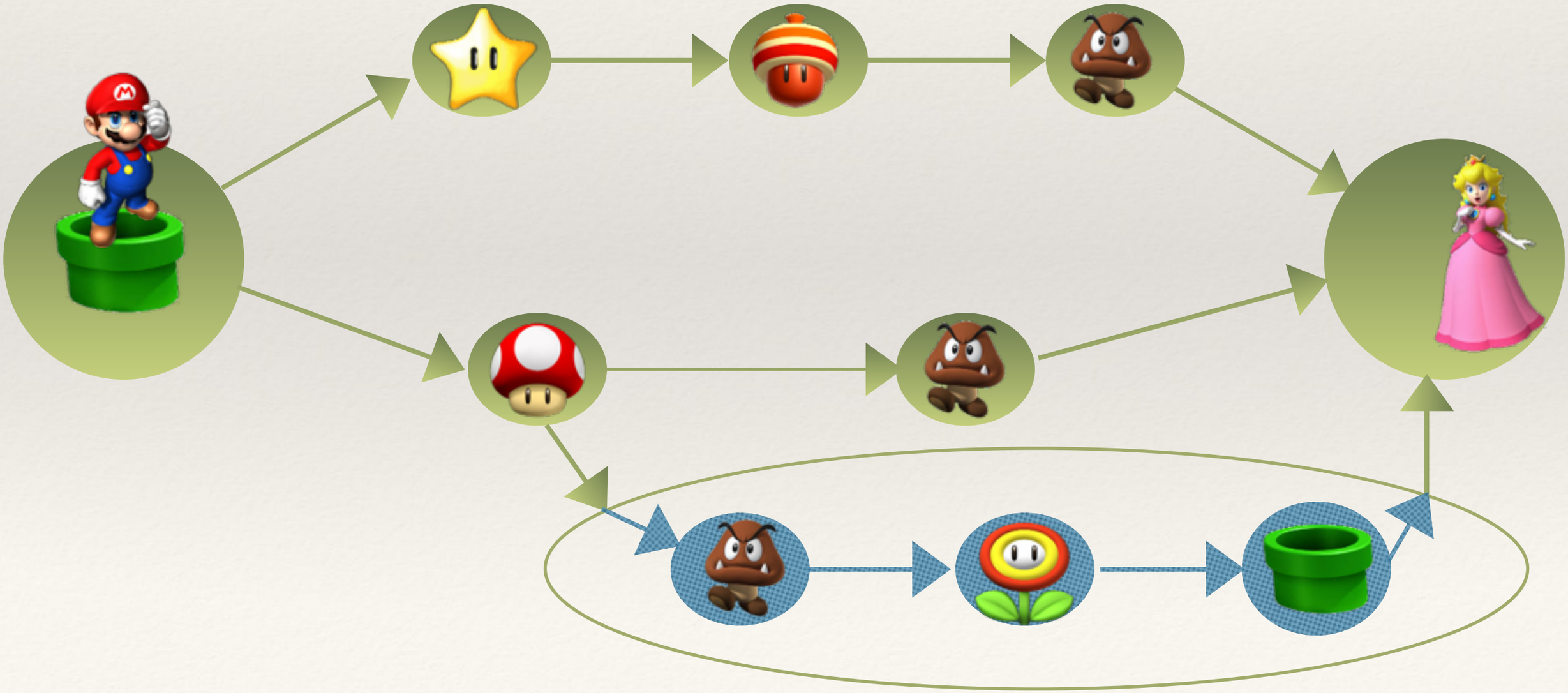
(2) the verification of incomplete models supports the replacement of ***complex parts*** *of the design with* ***incomplete parts*** *to reduce the verification time*



E(¬( ))U( )

Consideration of our Approach

*(3) in the **adaptive system** case, the verification of incomplete models allows to understand whether the system satisfies its requirements when the **different components are removed (plugged) into the running system***



Formalisms

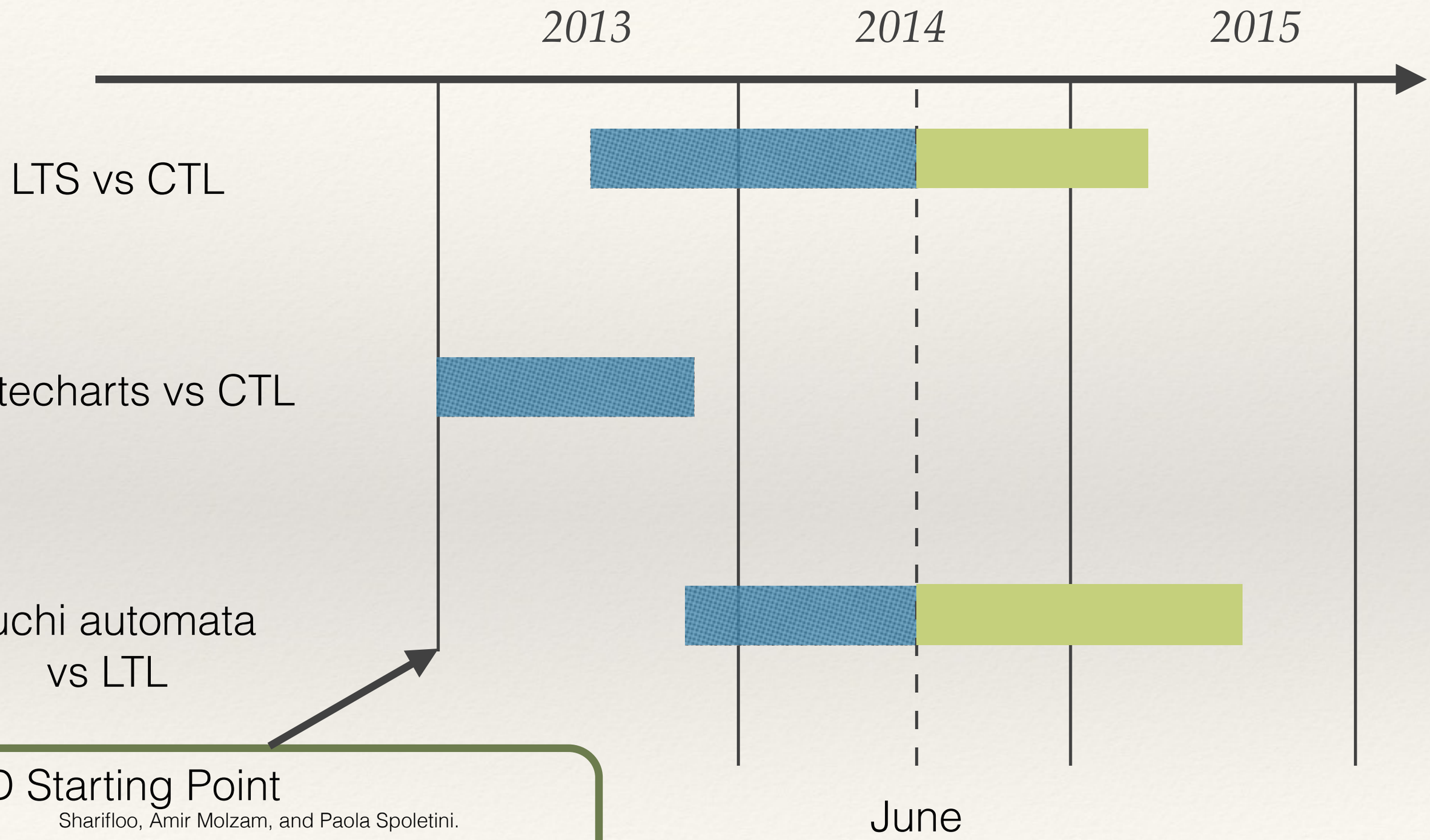
Specification	Requirements
Labeled Transition Systems (LTS)[1]	Computation Tree Logic (CTL)
Buchi automata	Linear-Time Temporal Logic (LTL)
Statechart[2,3]	Computation Tree Logic (CTL)
Markov Chains	Probabilistic CTL
Ambient Calculus	Ambient Logic

1 Sharifloo, Amir Molzam, and Paola Spoletini. "Lover: light-weight formal verification of adaptive systems at run time." Formal Aspects of Component Software.

2 Ghezzi, C., Menghi, C., Sharifloo, A., and Spoletini, P. "On Requirements Verification for Model Refinements." International Requirements Engineering Conference.

3 Ghezzi, C., Menghi, C., Sharifloo, A., and Spoletini, P. "On requirement verification for evolving Statecharts specifications." Requirements Engineering Journal

Research Progress



PhD Starting Point

Sharifloo, Amir Molzam, and Paola Spoletini.

"Lover: light-weight formal verification of adaptive systems at run time."

Formal Aspects of Component Software.

Thanks to

Carlo Ghezzi

Paola Spoletini

Amir Molzam Sharifloo

Questions ?