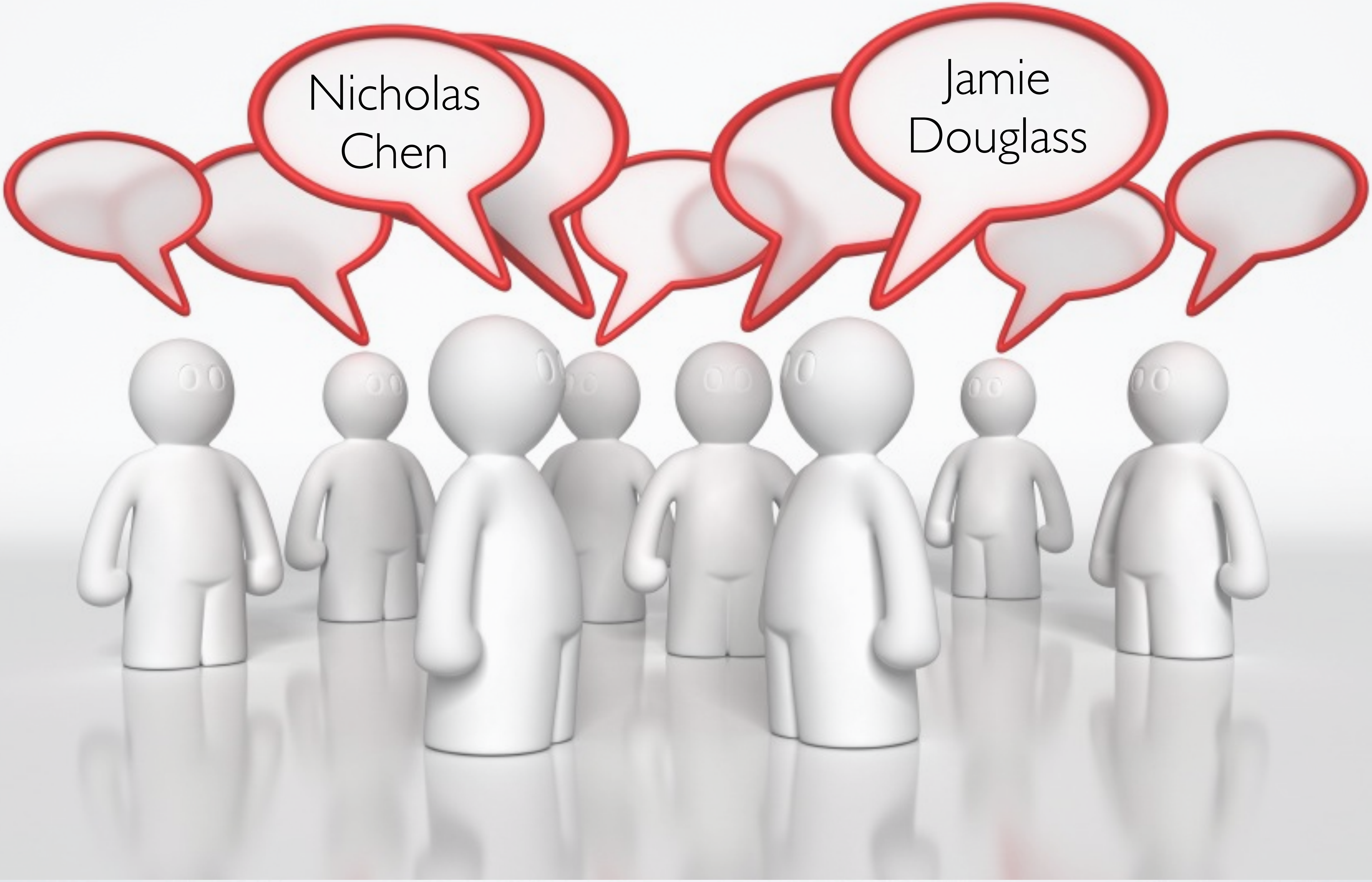
LANGUAGE OF LANGUAGES

www.languageoflanguages.org

LANGUAGE OF LANGUAGES

www.languageoflanguages.org



LANGUAGE OF LANGUAGES

www.languageoflanguages.org

Fortran

Lisp

Fortran

Lisp

Cobol

Fortran

Lisp

Cobol

Fortran


Lua
PHP
FSharp
Scala
Java
Groovy
Javascript
CSharp
Prolog
Haskell
Fortran
Perl
Octave
Bash
Go
Clojure
Lisp
Python
Ruby
C++
Objective-C
JRuby
Nasm
Erlang
Jython
Ocaml
Cobol
Scheme

UML
Perl
JRuby
Nasm
Erlang
Java
Javascript
Octave
Lua
CSharp
Bash
Go
PHP
Scala
Groovy
Objective-C
Jython
FSharp
Clojure
Lisp
Ocaml
Prolog
Python
Cobol
Haskell
Ruby
Fortran
C++
Scheme


UML
CSS
Lua
PHP
FSharp
Scala
Java
Groovy
Javascript
Prolog
CSharp
Haskell
Fortran
Perl
Octave
Bash
Go
Clojure
Lisp
Python
Ruby
C++
Objective-C
JRuby
Nasm
Erlang
Jython
Ocaml
Cobol
Scheme

UML LaTeX
CSS Java Perl
Lua Javascript CSharp Octave JRuby Nasm Erlang
PHP Scala Groovy Bash Go
FSharp Clojure Objective-C Jython
Lisp Ocaml
Prolog Python Cobol
Haskell Ruby
Fortran C++ Scheme

UML LaTeX
CSS Java Perl
Javascript Octave JRuby Nasm Erlang
Lua
CSharp
PHP Scala Groovy Bash Go
Objective-C
Jython
FSharp Clojure
Lisp Ocaml
Prolog Python Cobol
Haskell Ruby
Fortran XML C++ Scheme

UML LaTeX
CSS Java Perl
Javascript Octave JRuby Nasm Erlang
Lua CSharp
PHP Scala Bash Go
Groovy Objective-C Jython
FSharp Clojure
Lisp Ocaml
Prolog
Python Cobol
Haskell Ruby Regex
Fortran XML C++ Scheme

UML LaTeX Graphviz

CSS Java Perl JRuby Nasm Erlang

Lua Javascript CSharp Octave

PHP Scala Groovy Bash Go

FSharp Clojure Objective-C Jython

Lisp Ocaml

Prolog Python Cobol

Haskell Ruby Regex

Fortran XML C++ Scheme

LANGUAGES TYPICALLY USED

LANGUAGES TYPICALLY USED

LANGUAGES TYPICALLY USED

Models

Matlab	Modelica
Simulink	...

LANGUAGES TYPICALLY USED

Models	
Matlab	Modelica
Simulink	...

Architecture	
AADL	SysML
UML	...

LANGUAGES TYPICALLY USED

Models		Architecture		Software	
Matlab	Modelica	AADL	SysML	C/C++	Fortran
Simulink	...	UML	...	Java	...

LANGUAGES TYPICALLY USED

Models

Matlab	Modelica
Simulink	...

SPECIALIZED DOMAINS

Models

Matlab	Modelica
Simulink	...

SPECIALIZED DOMAINS

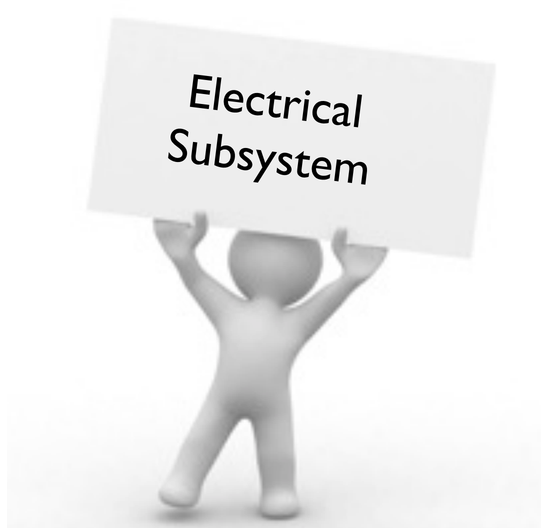
Models

Matlab
Modelica
Simulink
...

SPECIALIZED DOMAINS

Models

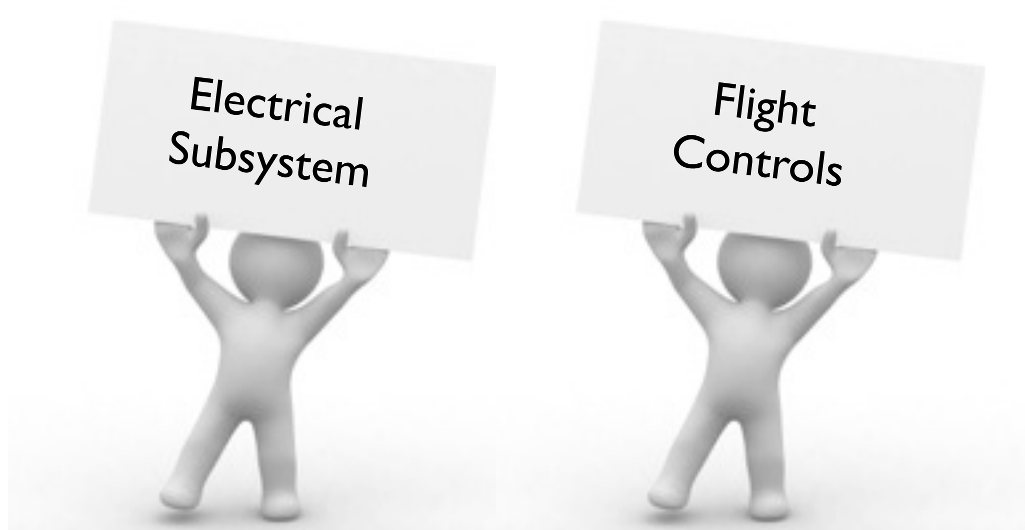
Matlab	Modelica
Simulink	...



SPECIALIZED DOMAINS

Models

Matlab
Modelica
Simulink
...



SPECIALIZED DOMAINS

Models

Matlab

Modelica

Simulink

...

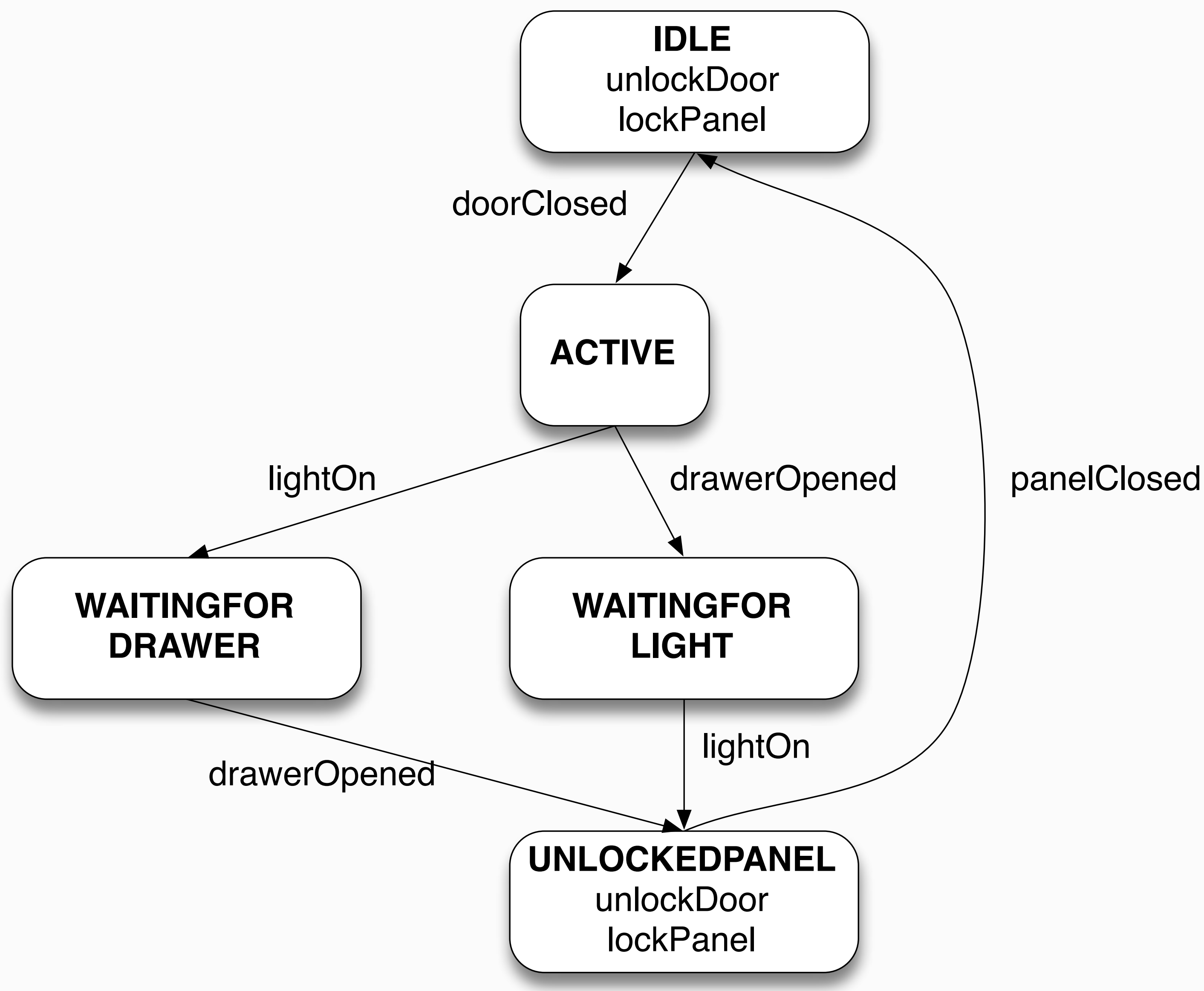
SPECIALIZED DOMAINS

Models

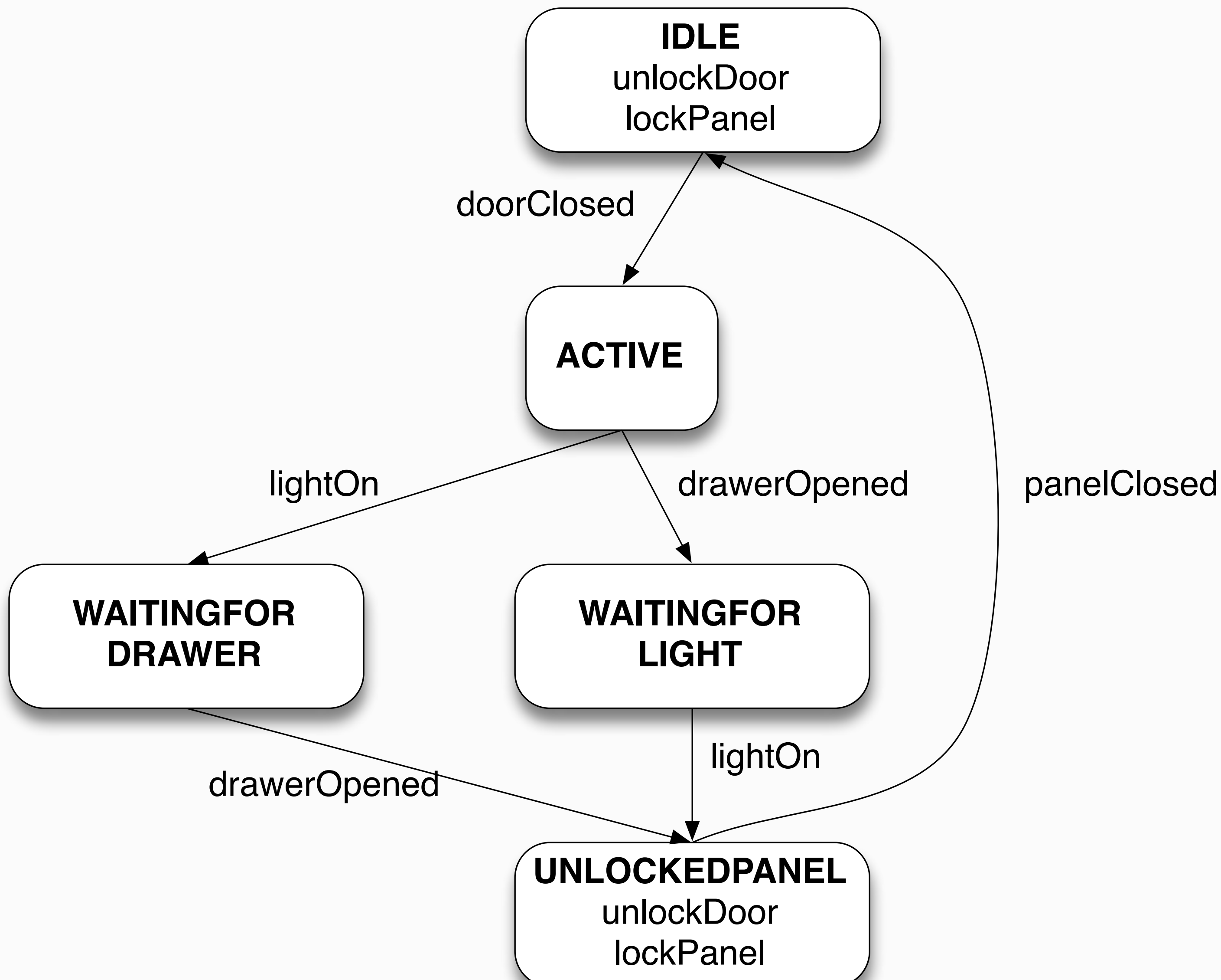
Matlab
Modelica
Simulink
...

EXAMPLE MODEL

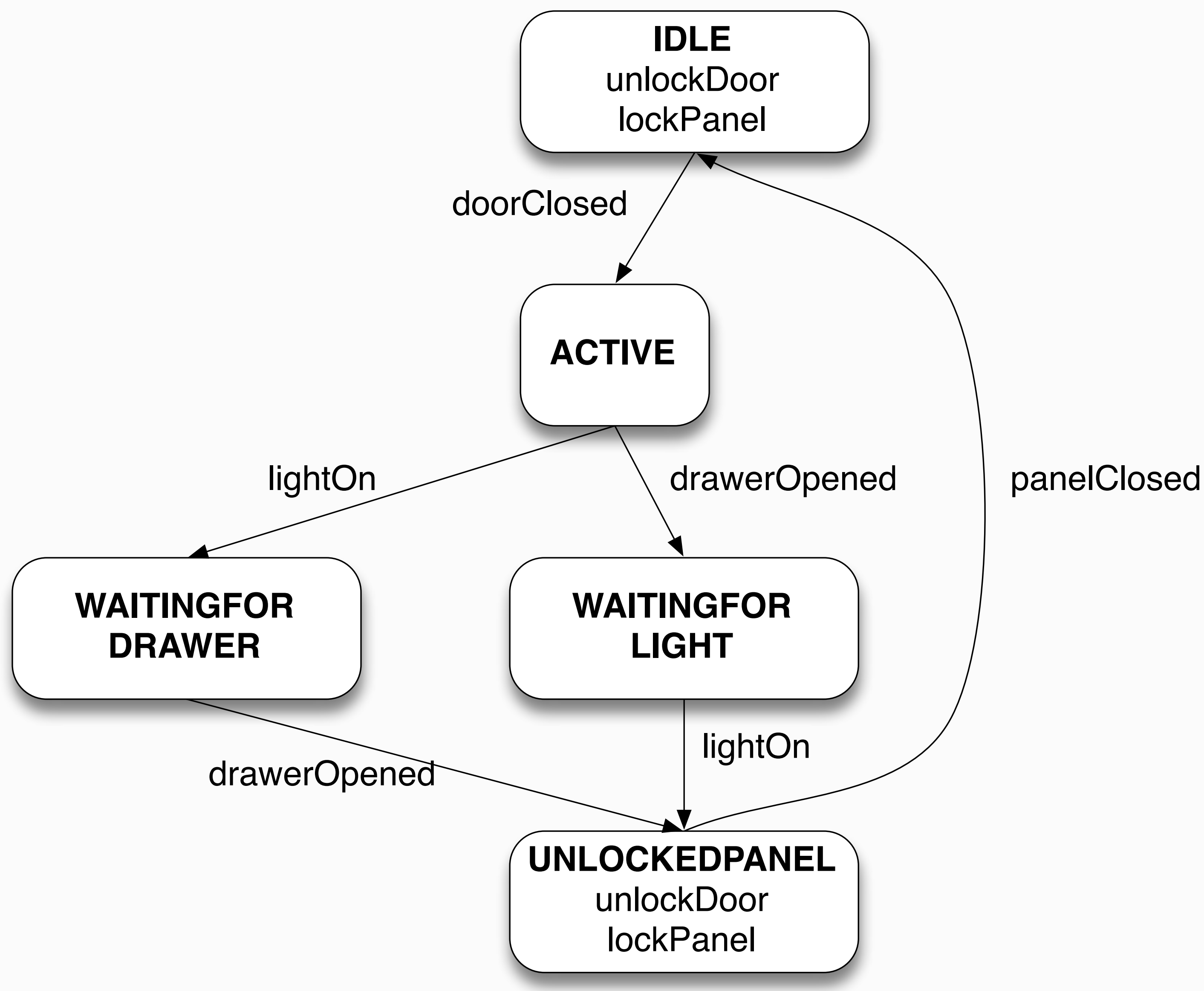
EXAMPLE MODEL



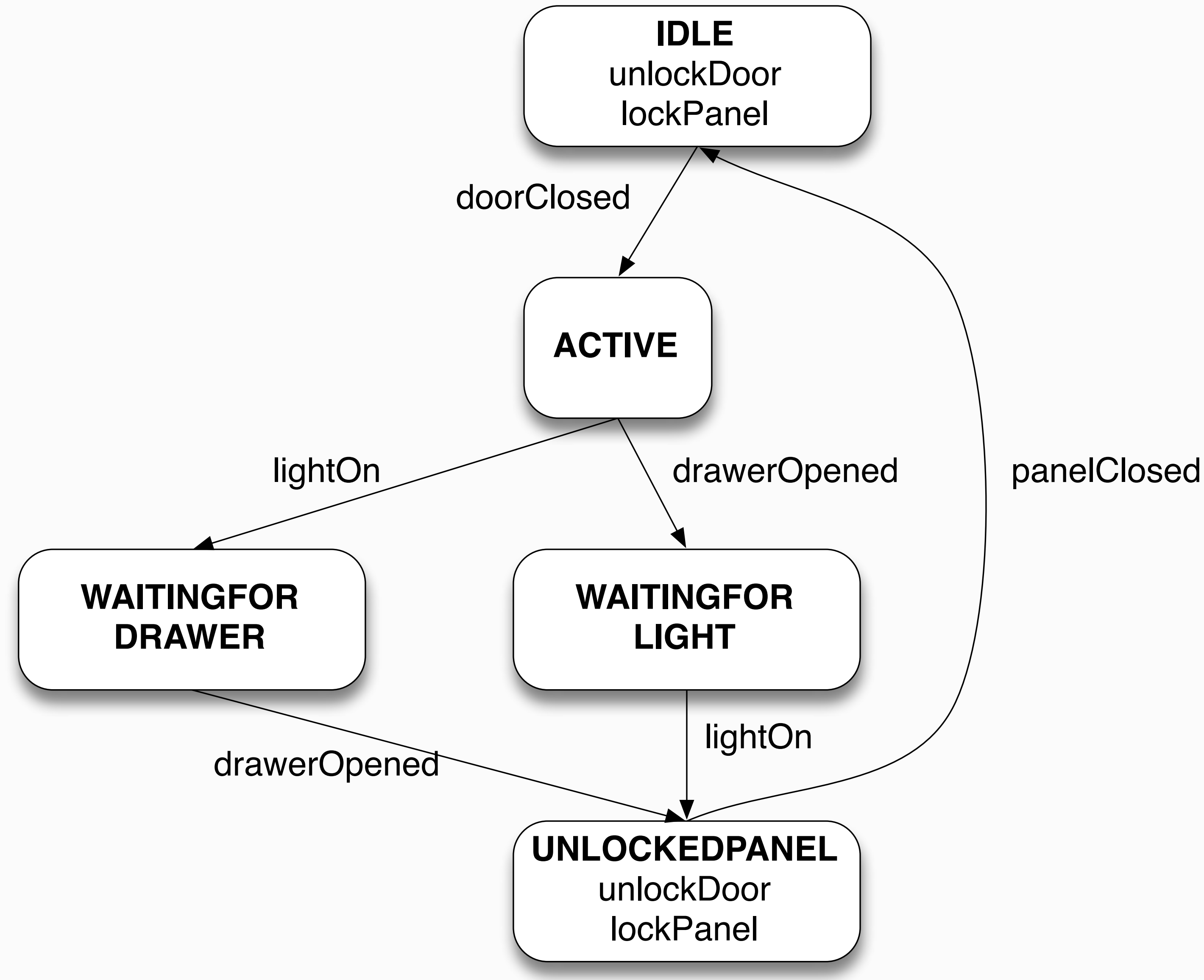
EXAMPLE MODEL



FOR THE PROGRAMMER

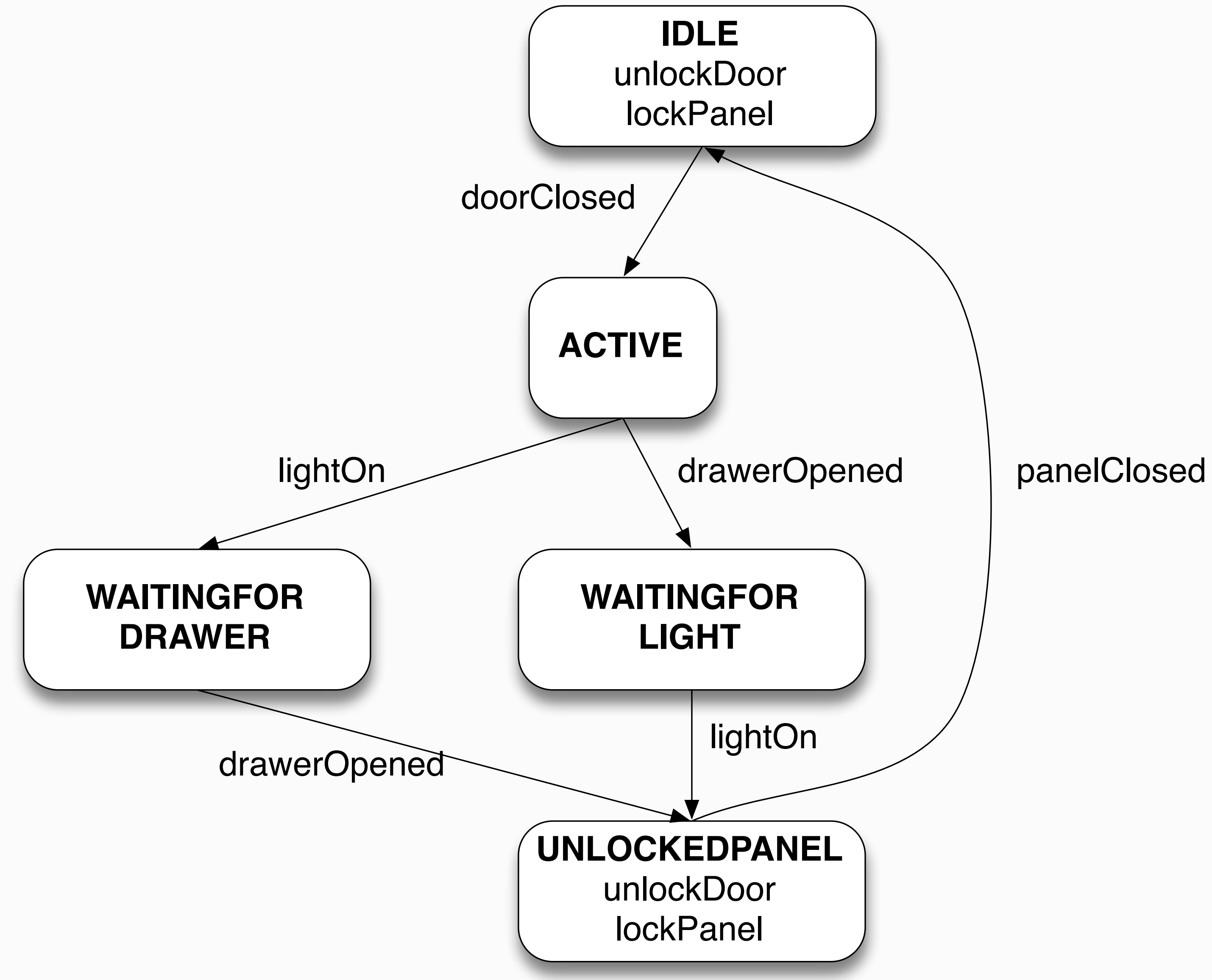


FOR THE PROGRAMMER



FOR THE PROGRAMMER

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```



FOR THE ENGINEER

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

FOR THE ENGINEER

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

FOR THE ENGINEER

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

FOR THE SYSTEMS ANALYST

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

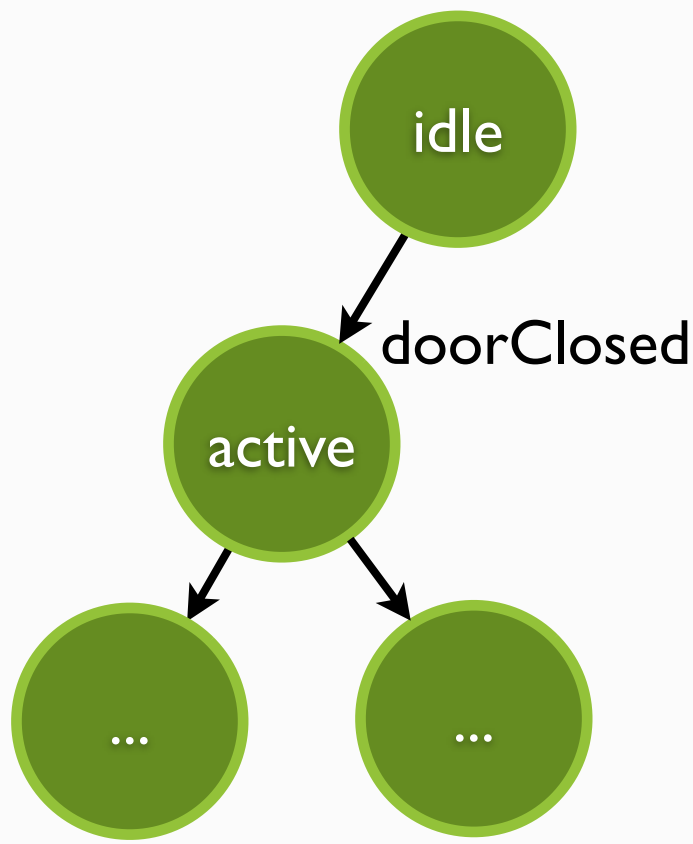
FOR THE SYSTEMS ANALYST

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

FOR THE SYSTEMS ANALYST

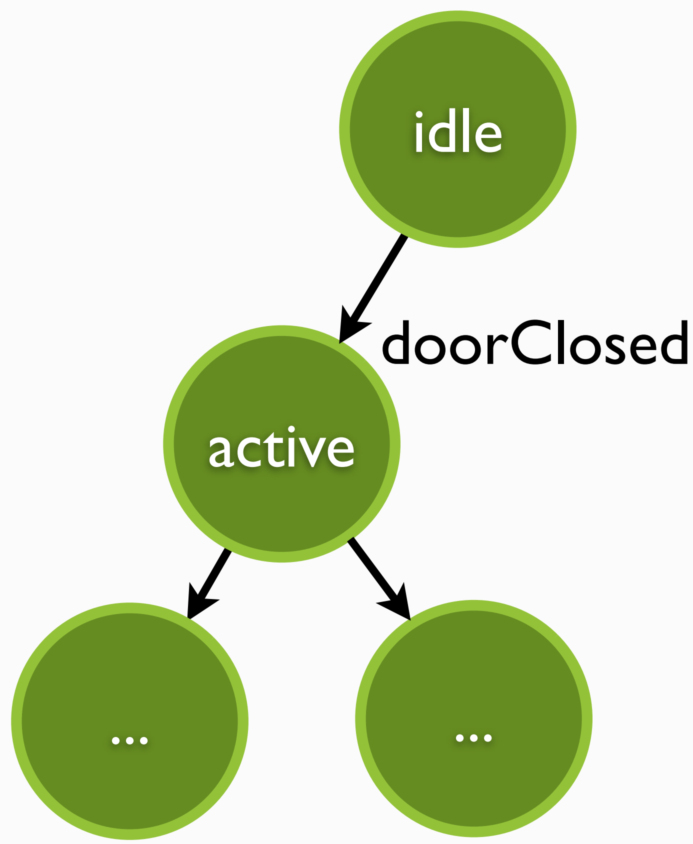
```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```




```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

KEEP IN SYNC

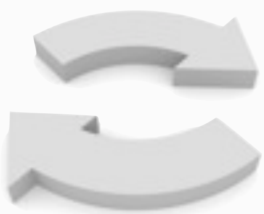
```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

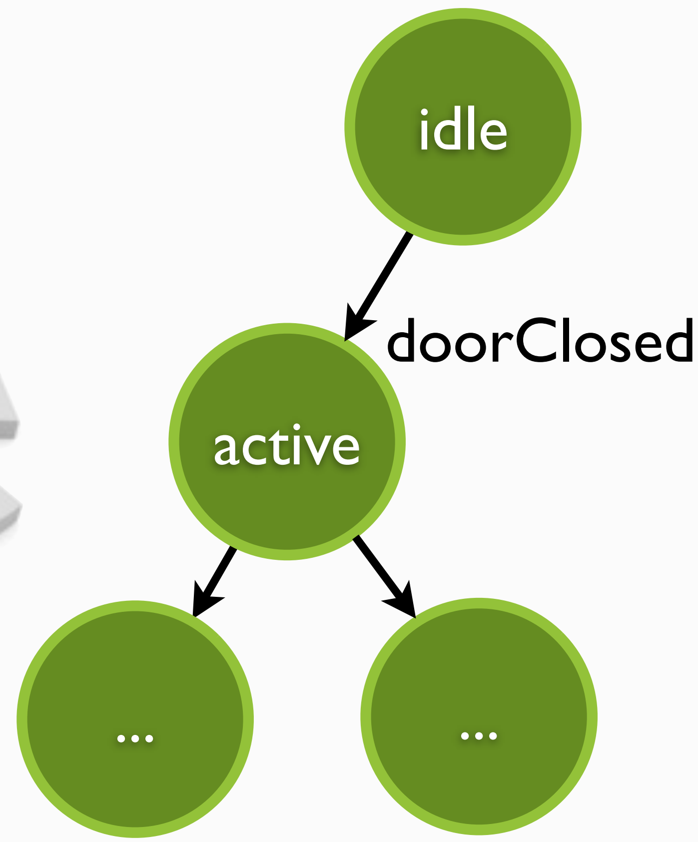



```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

KEEP IN SYNC

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```

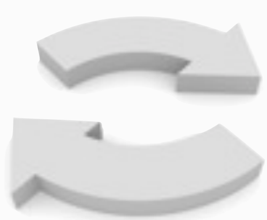


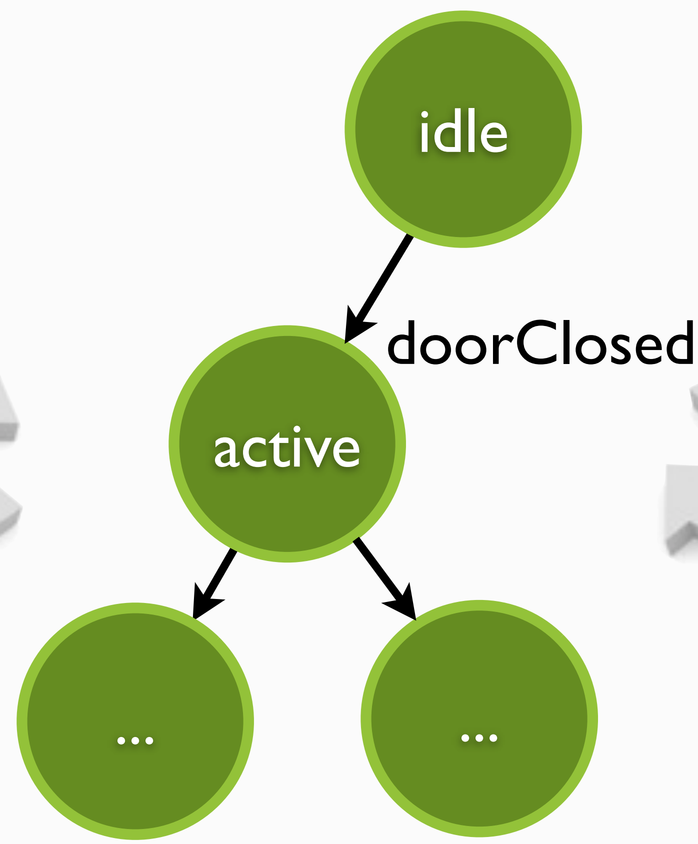


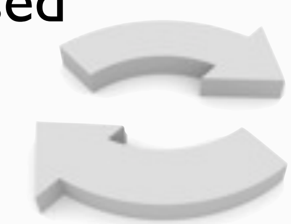

```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

KEEP IN SYNC

```
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");
```







```
events
    doorClosed
    drawerOpened
    ...
end

commands
    unlockPanel
    lockPanel
    ...
end

state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
```

LANGUAGE OF LANGUAGES

LANGUAGE OF LANGUAGES

Experimental language workbench that *embraces* the use of multiple notations (textual and graphical)

LANGUAGE OF LANGUAGES

Experimental language workbench that *embraces* the use of multiple notations (textual and graphical) *for flexible development*

LANGUAGE OF LANGUAGES

LANGUAGE OF LANGUAGES

Language Workbench: IDE for *convenient* language experimentation (creating, editing, translating)

LANGUAGE OF LANGUAGES

Language Workbench: IDE for *convenient* language experimentation (creating, editing, translating)

ROADMAP

ROADMAP

- Support a mix of textual and graphical languages.
- Support parsing as well as projecting
- Minimal paradigm with Language Elements, Concepts, Language Definitions.
- Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for *plug-n-play*
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs

ROADMAP

- ✔ Support a mix of textual and graphical languages.
- Support parsing as well as projecting
- Minimal paradigm with Language Elements, Concepts, Language Definitions.
- Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for *plug-n-play*
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs

ROADMAP

- ✔ Support a mix of textual and graphical languages.
- ✔ Support parsing as well as projecting
- Minimal paradigm with Language Elements, Concepts, Language Definitions.
- Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for *plug-n-play*
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs

ROADMAP

- ✔ Support a mix of textual and graphical languages.
- ✔ Support parsing as well as projecting
- ✔ Minimal paradigm with Language Elements, Concepts, Language Definitions.
- Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for *plug-n-play*
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs

ROADMAP

✔ • Support a mix of textual and graphical languages.

✔ • Support parsing as well as projecting

✔ • Minimal paradigm with Language Elements, Concepts, Language Definitions.

• Support for outline, syntax coloring, code completion, etc

• Support for language debugging

• Web based app (like lively kernel)

• Community repository of Concepts for *plug-n-play*

• Fine-grained version control based on concepts

• Meta-circularity: LoLs is implemented in LoLs

SMALL TASTE

SMALL TASTE

SMALL TASTE

- Calculation

SMALL TASTE

- Calculation
- Western Math

SMALL TASTE

- Calculation
- Western Math
- Roman Numerals

SMALL TASTE

- Calculation
- Western Math
- Roman Numerals
- Stack Machine

HOW IT WORKS

Concepts

CALCULATOR EXAMPLE

$$3 + 4$$

CALCULATOR EXAMPLE

❶ Numbers

$$3 + 4$$

CALCULATOR EXAMPLE

❶ Numbers
❷ Addition

$$3 + 4$$

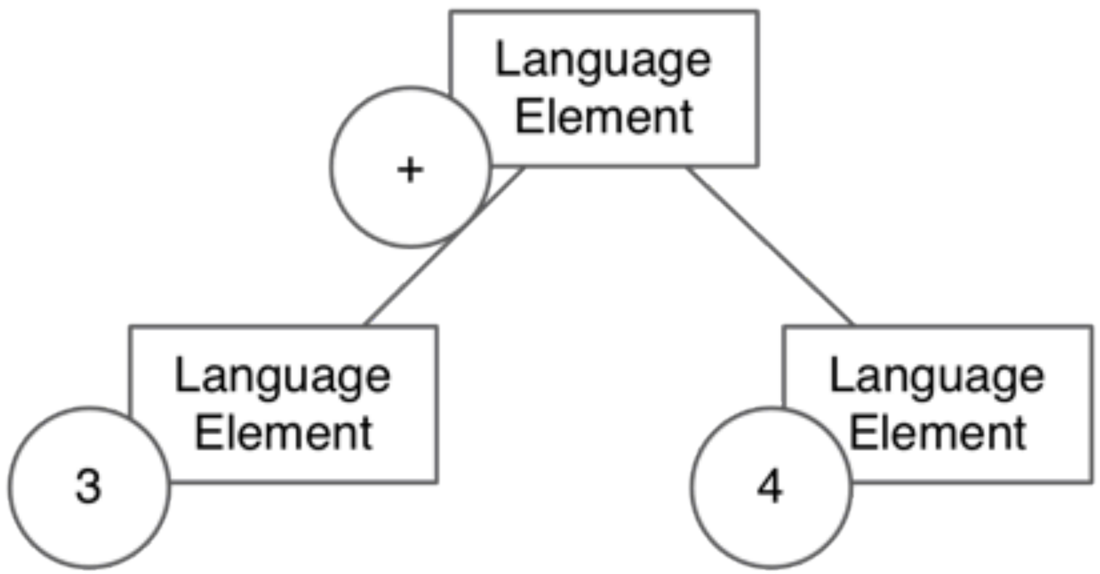
$$3 + 4$$

3 + 4

Parse!

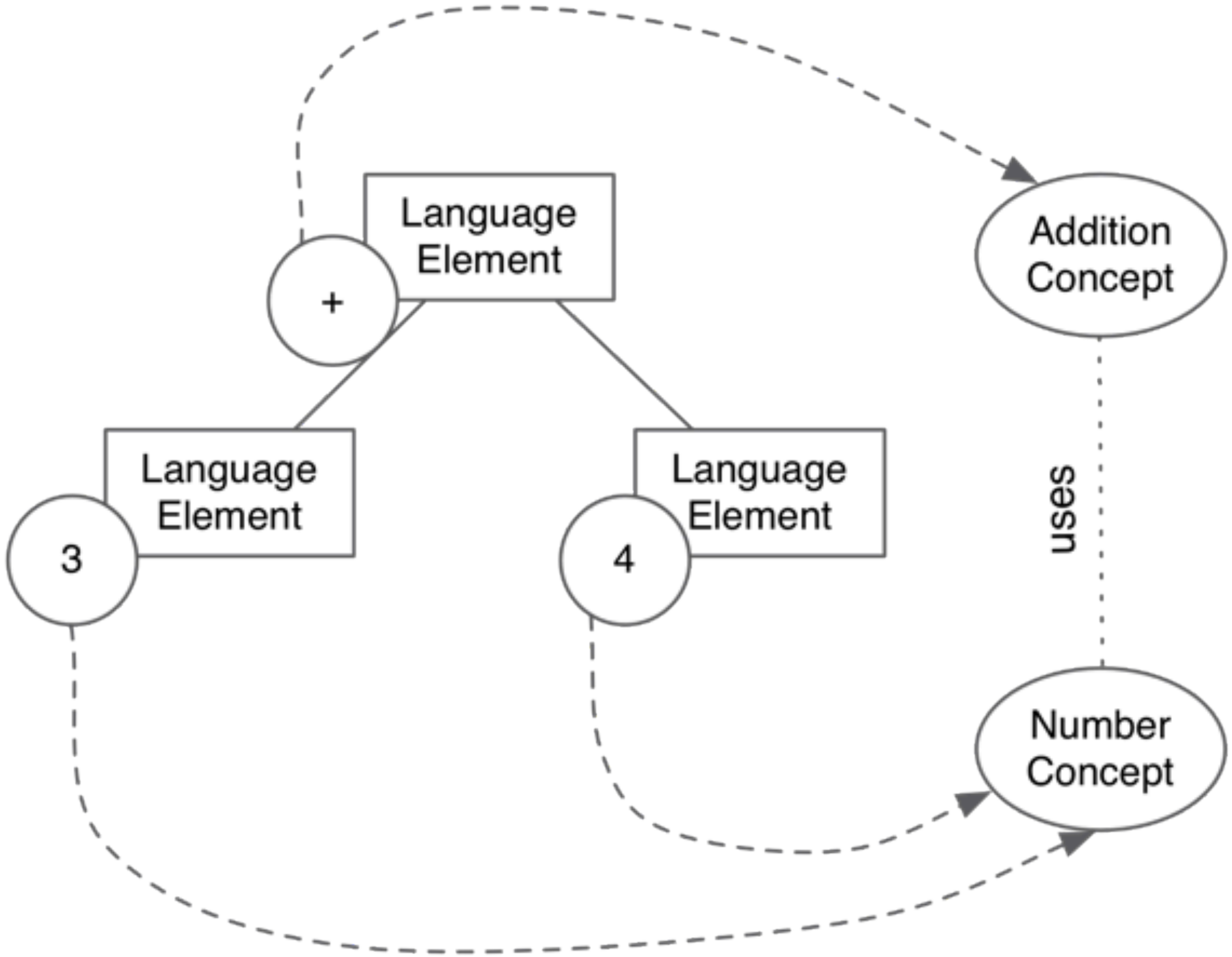
TREE REPRESENTATION

Language Element Tree (LET)

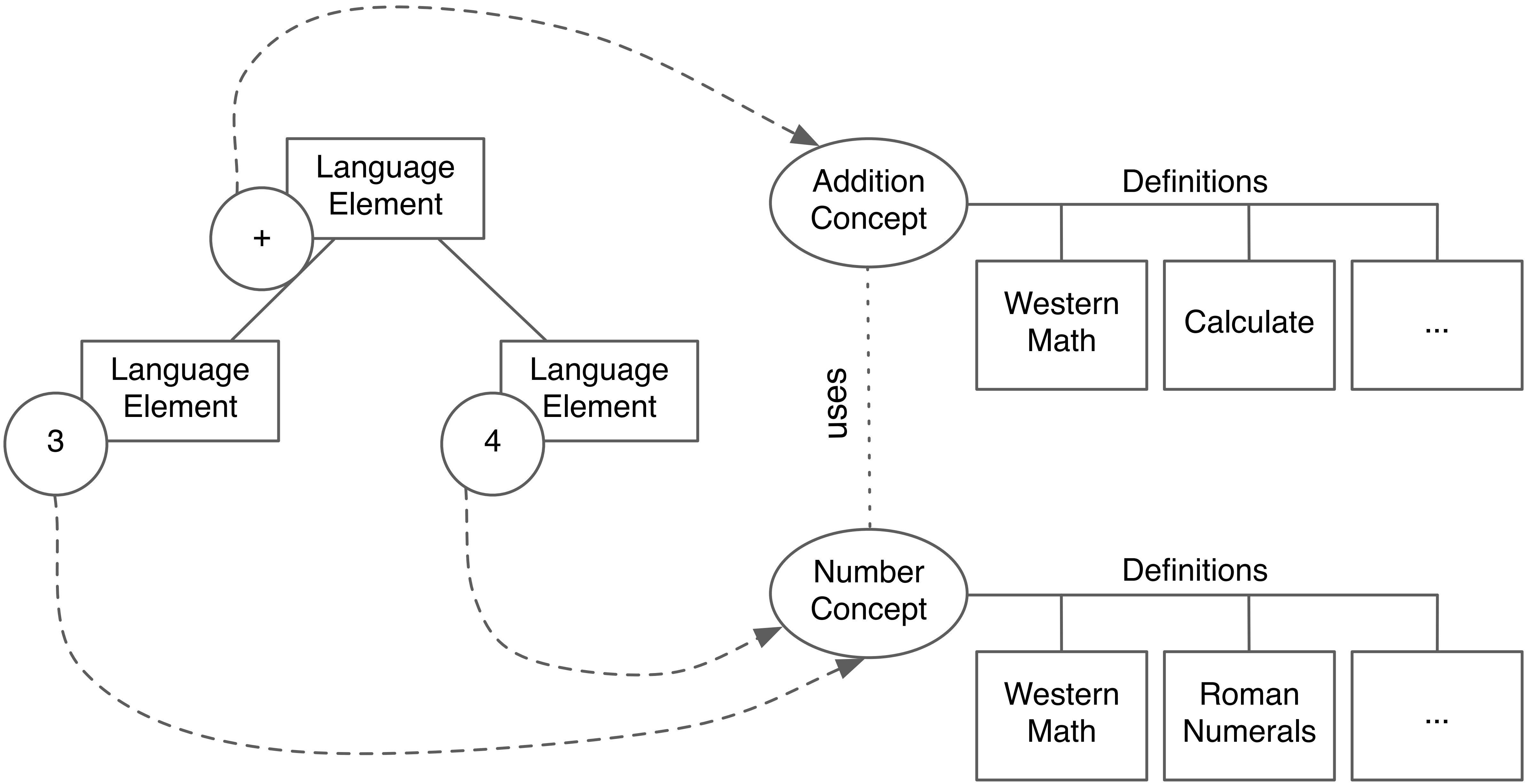


TREE REPRESENTATION

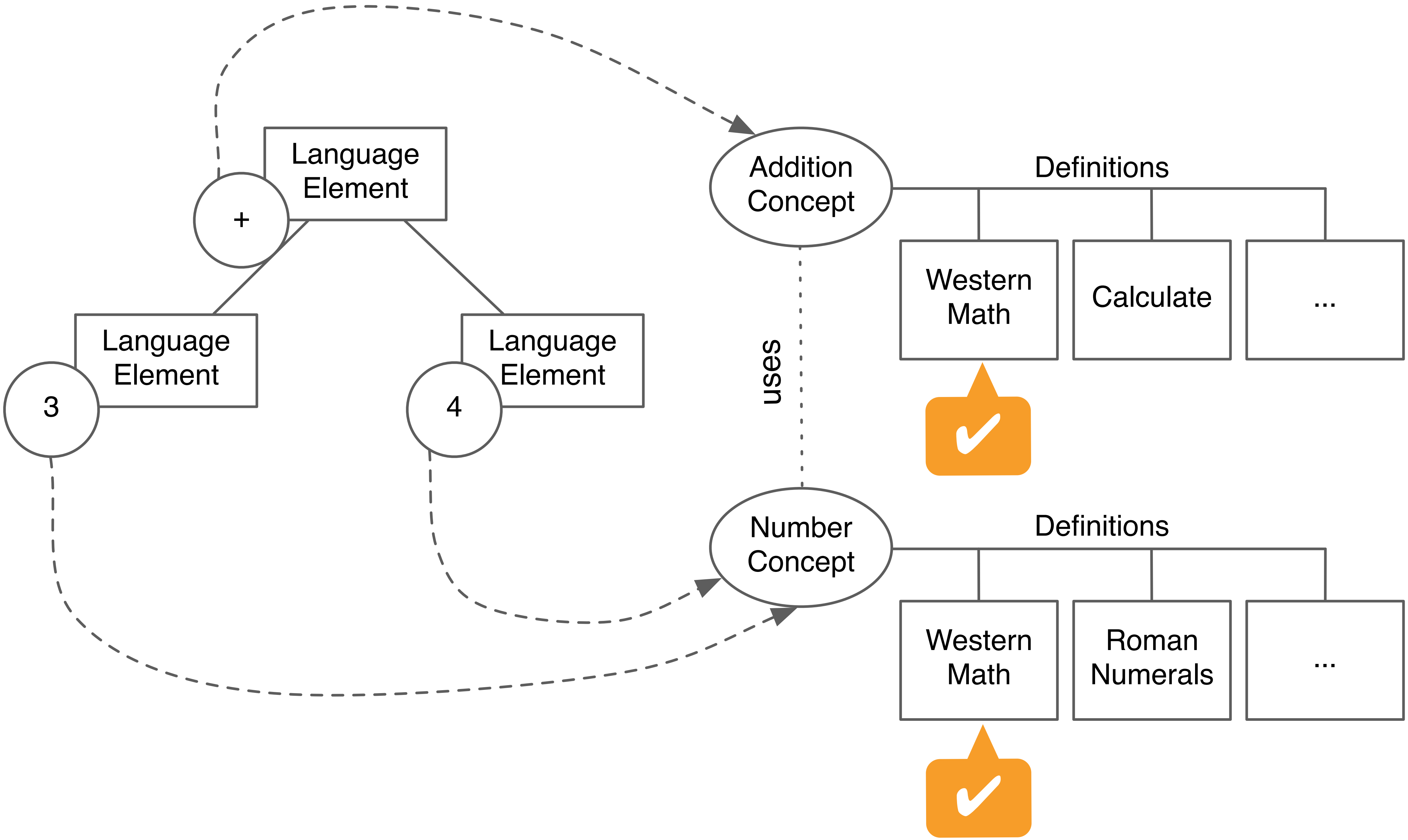
Language Element Tree (LET)



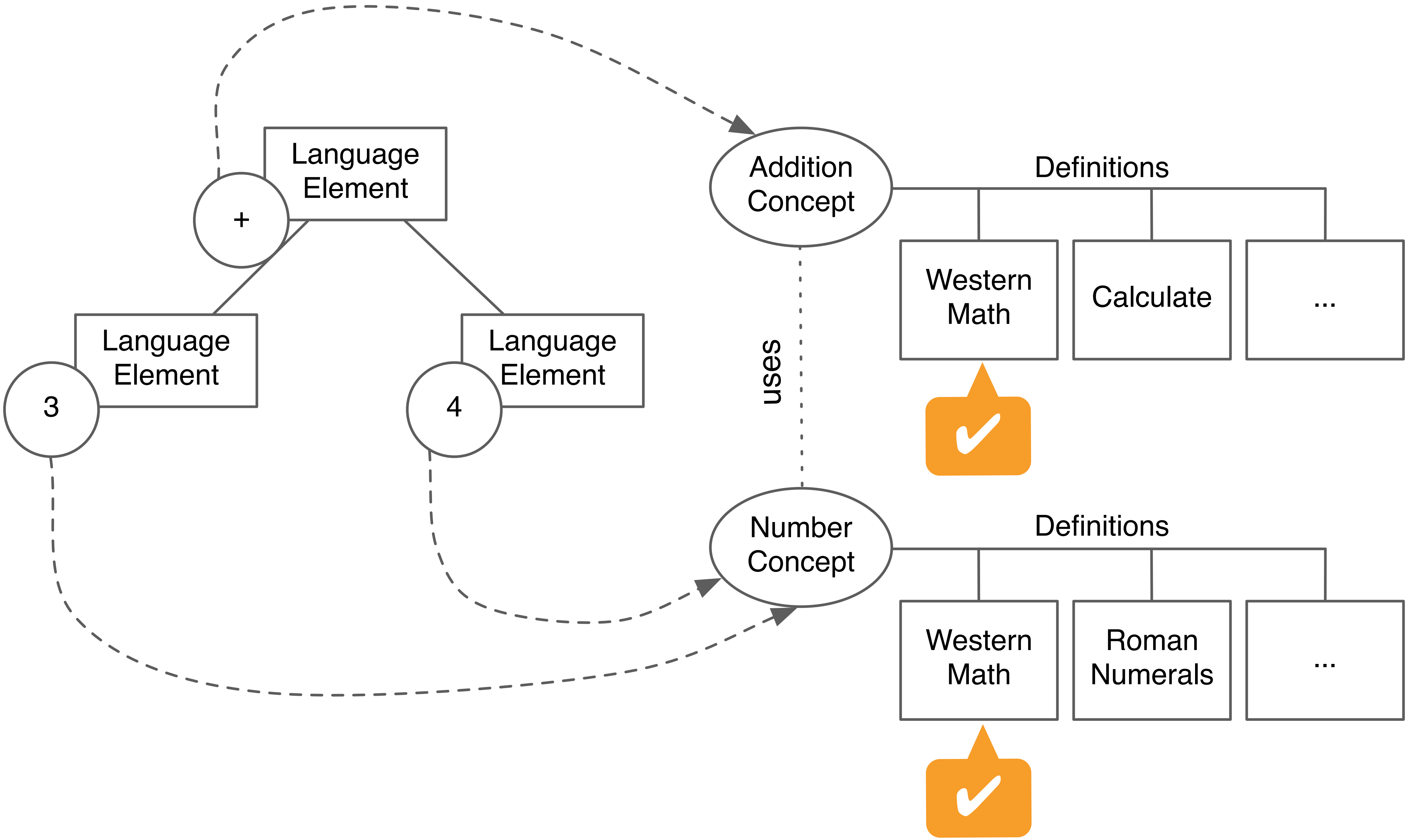
ATTACH CONCEPTS



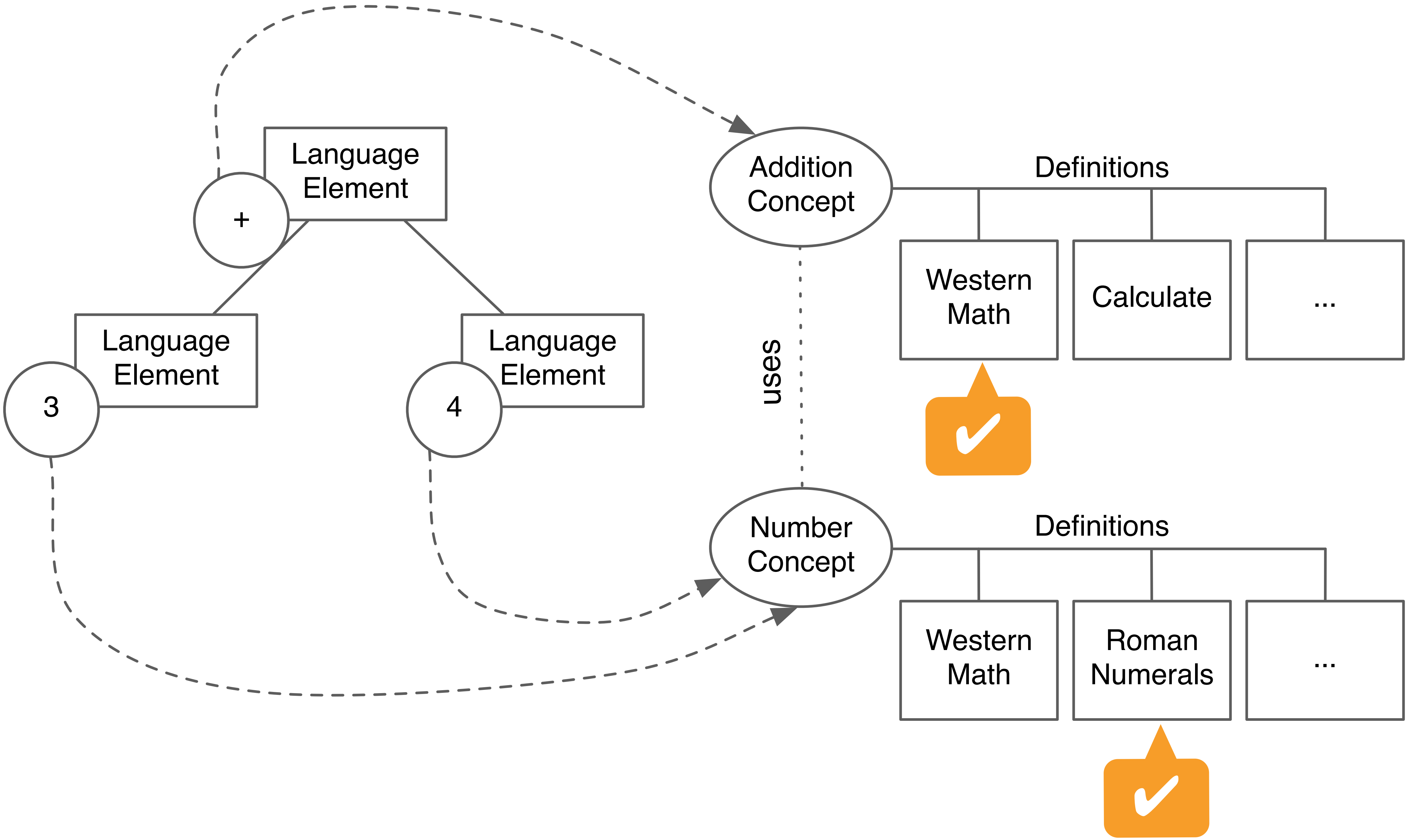
ATTACH DEFINITIONS



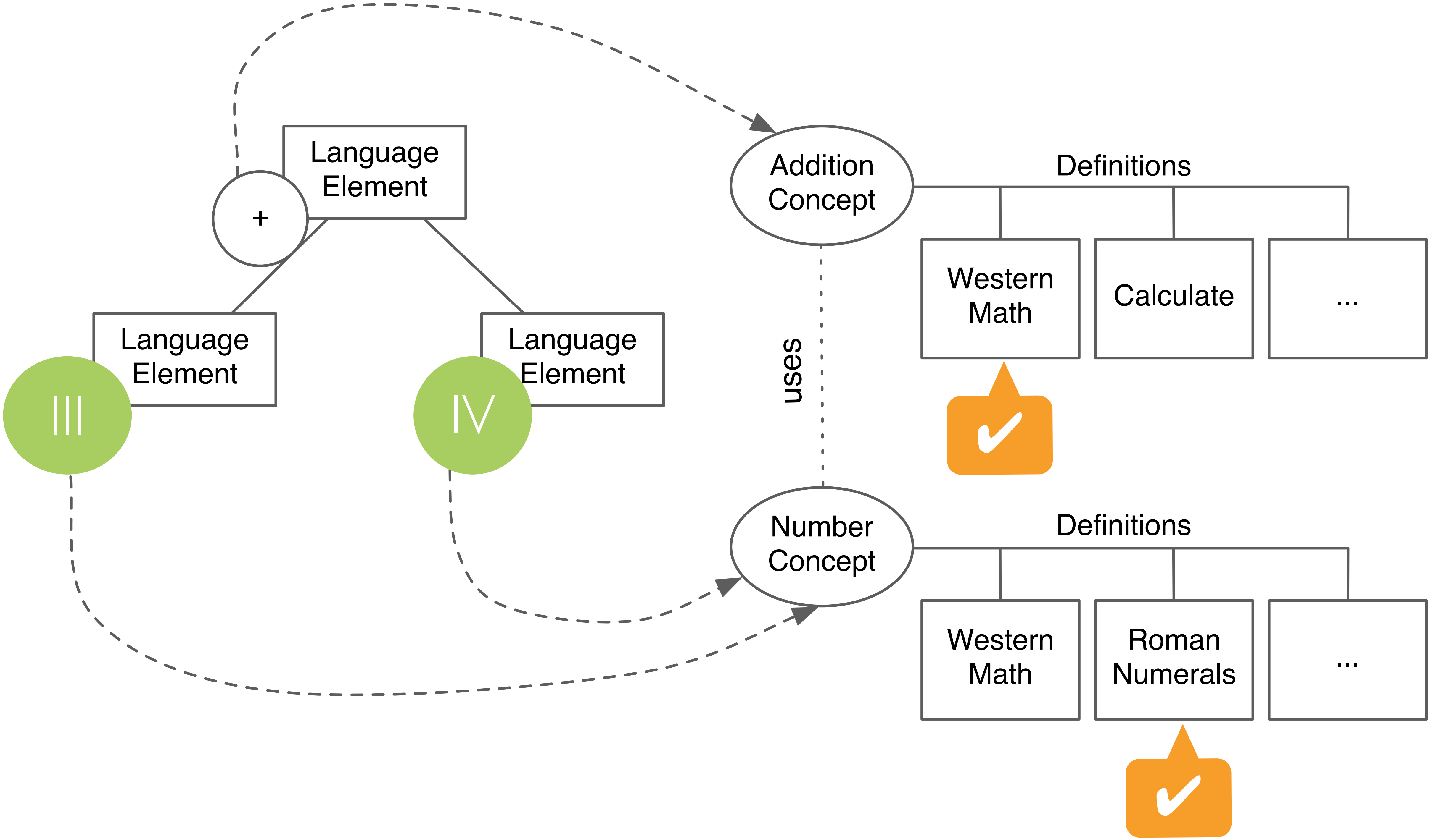
ATTACH DEFINITIONS



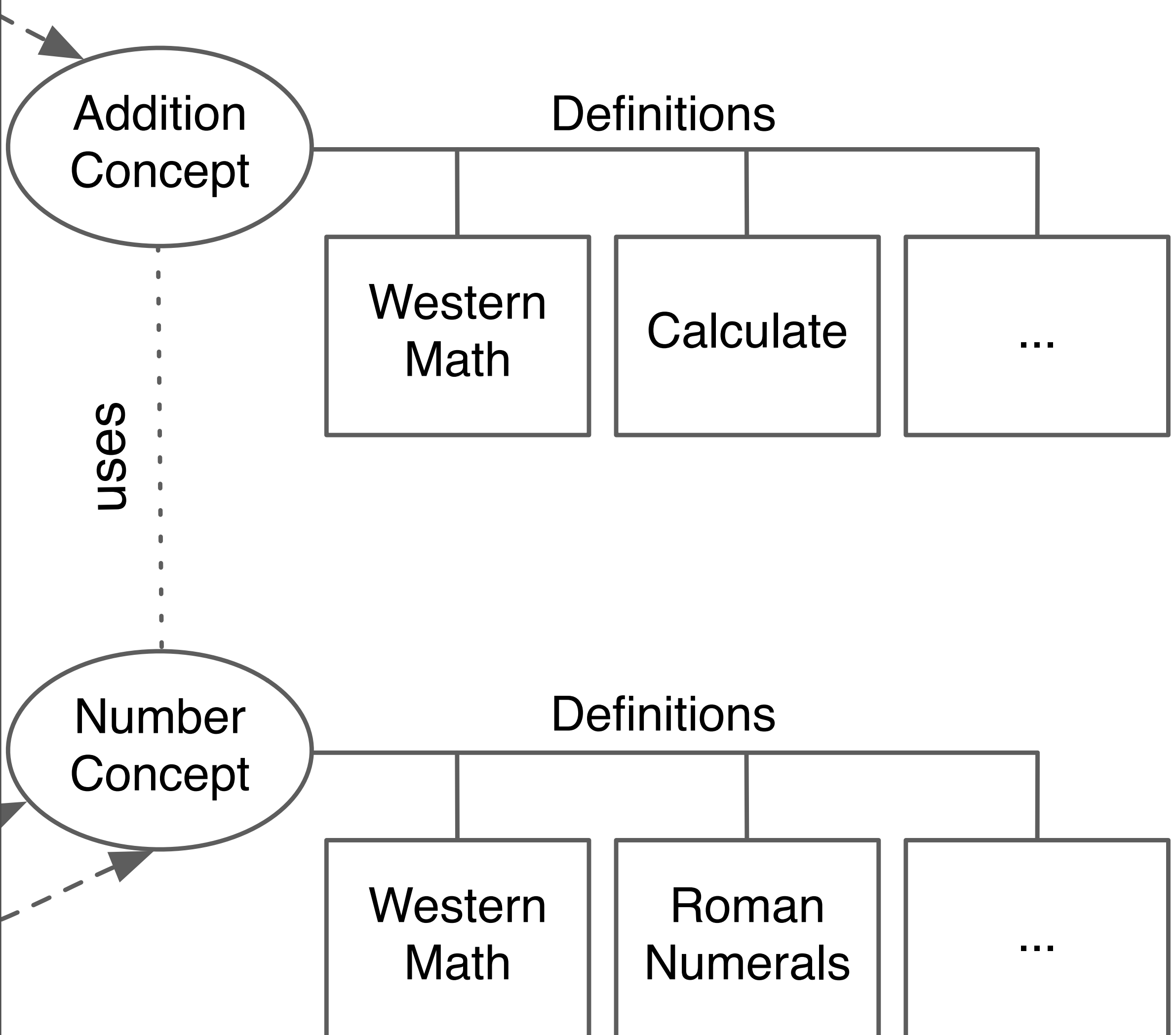
SWITCH DEFINITIONS



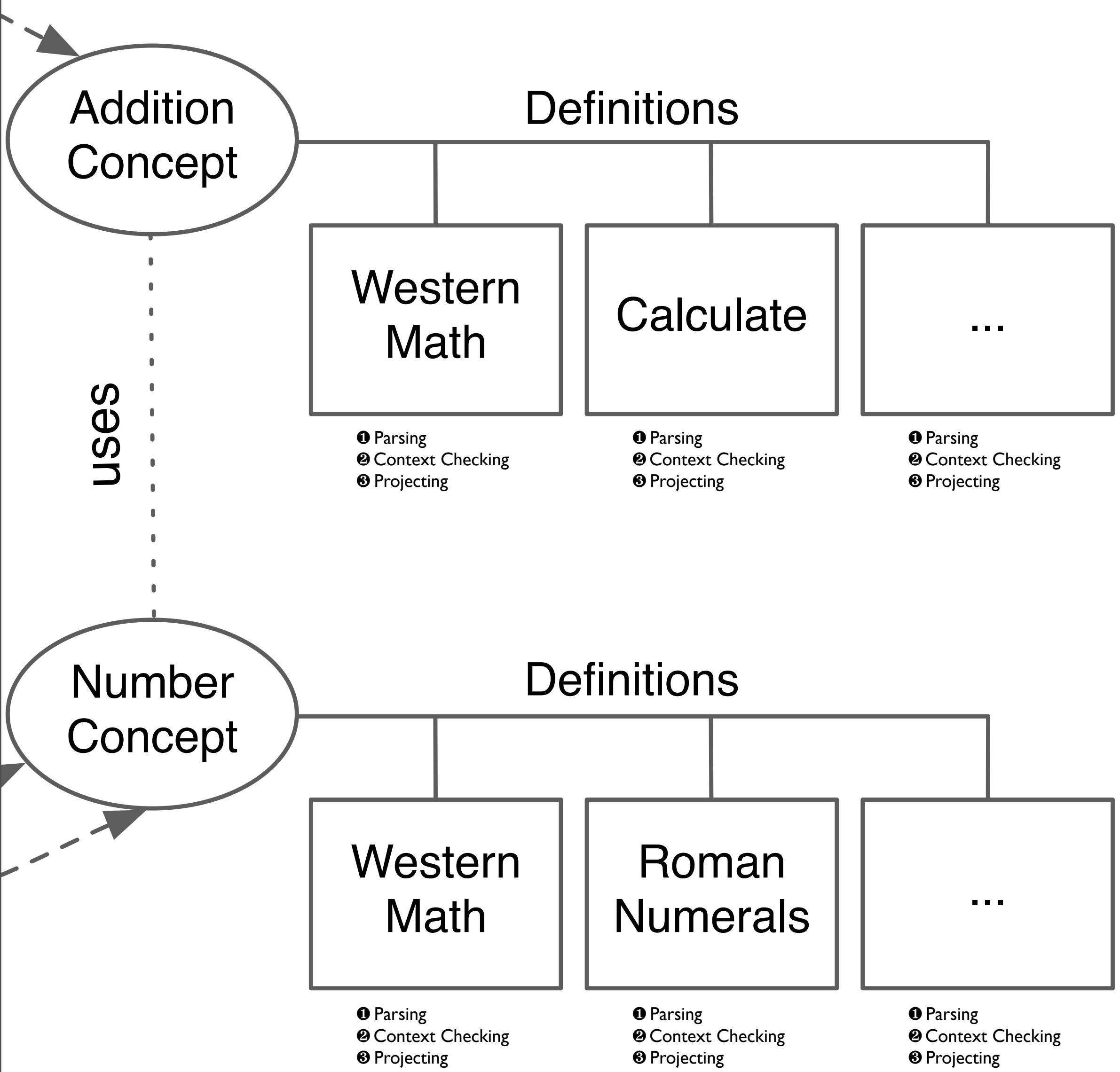
SWITCH DEFINITIONS



SWITCH DEFINITIONS

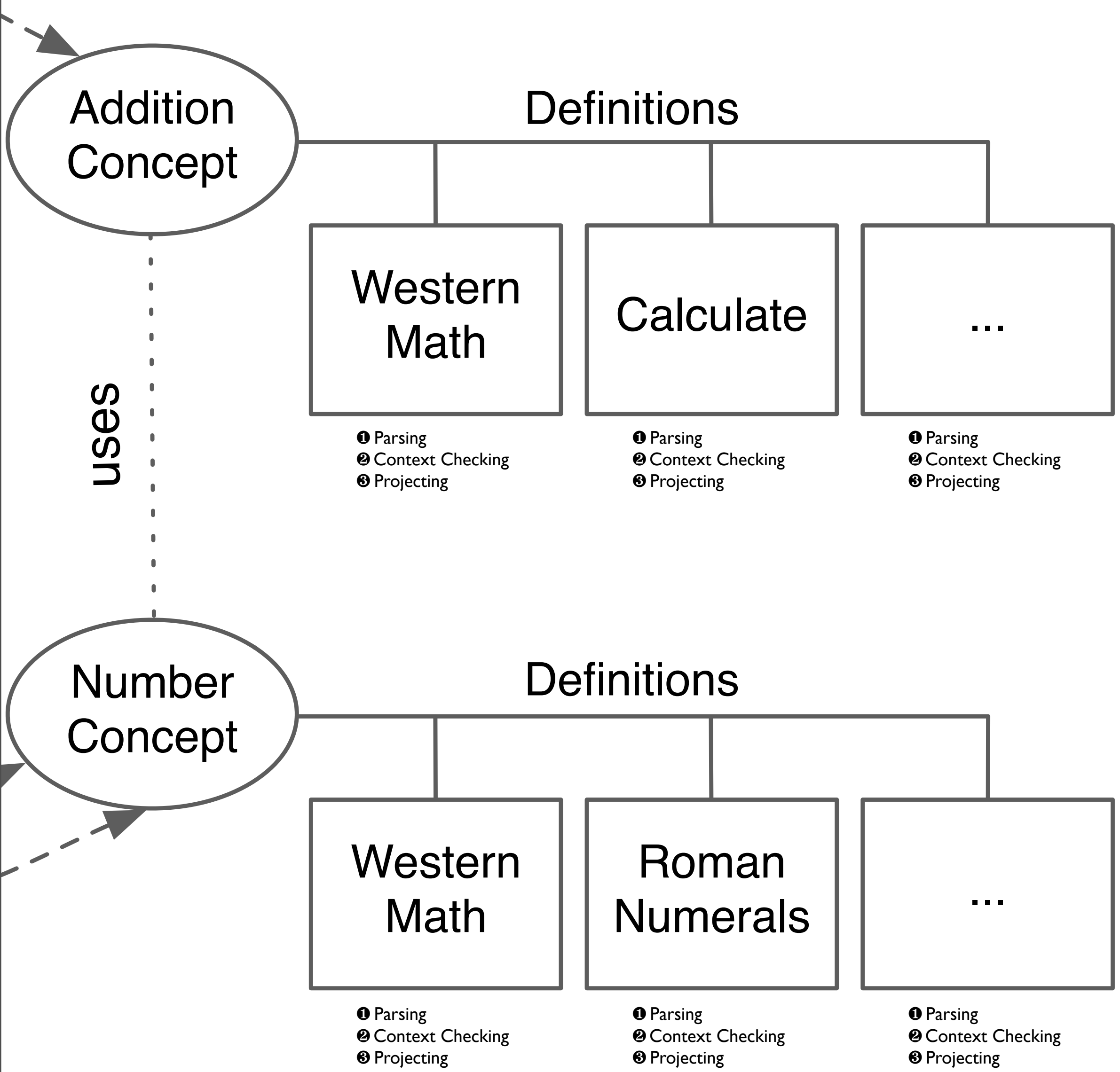


LANGUAGE DEFINITIONS



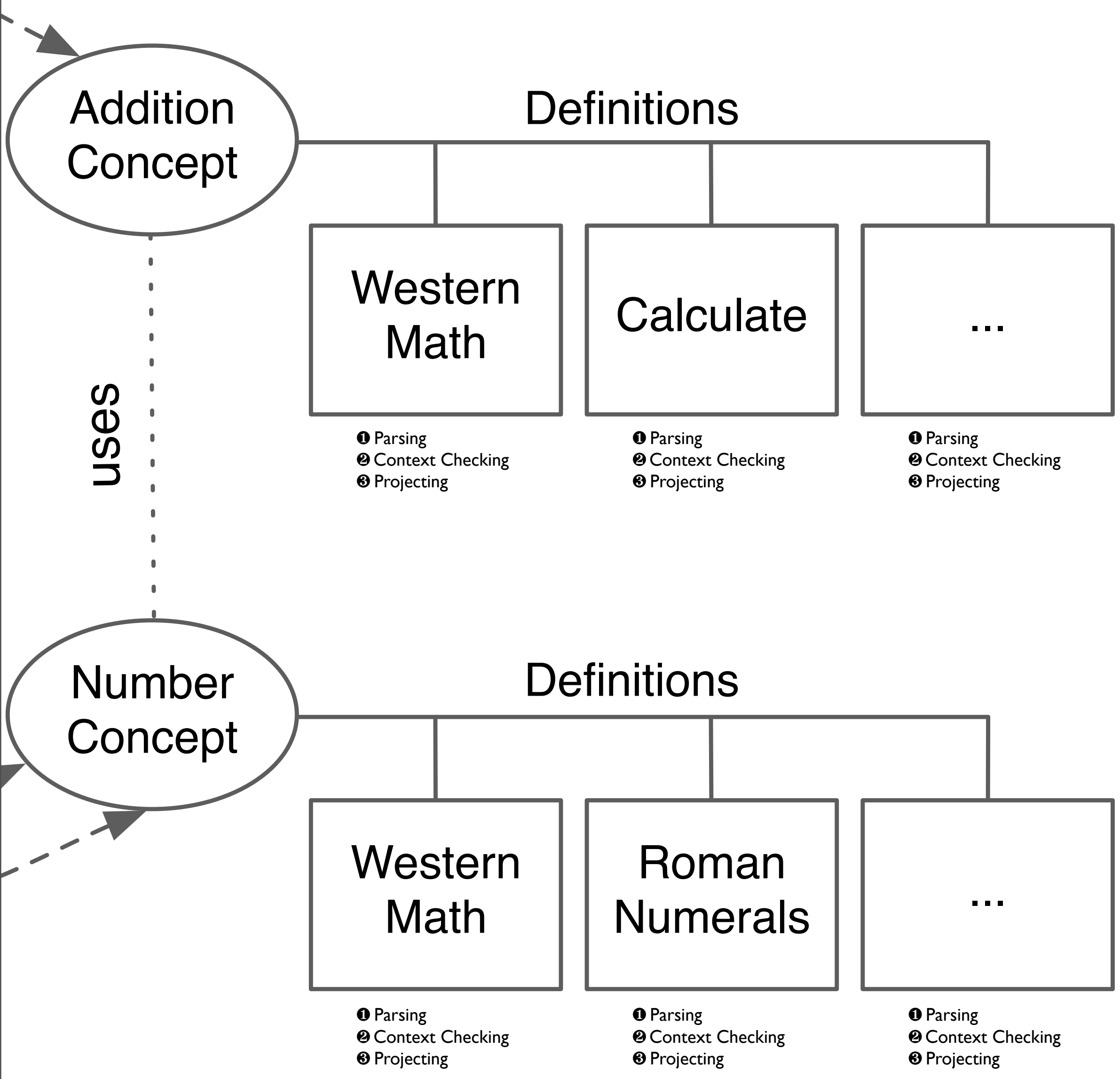
LANGUAGE DEFINITIONS

❶Parsing
❷Context Checking
❸Projecting



LANGUAGE DEFINITIONS

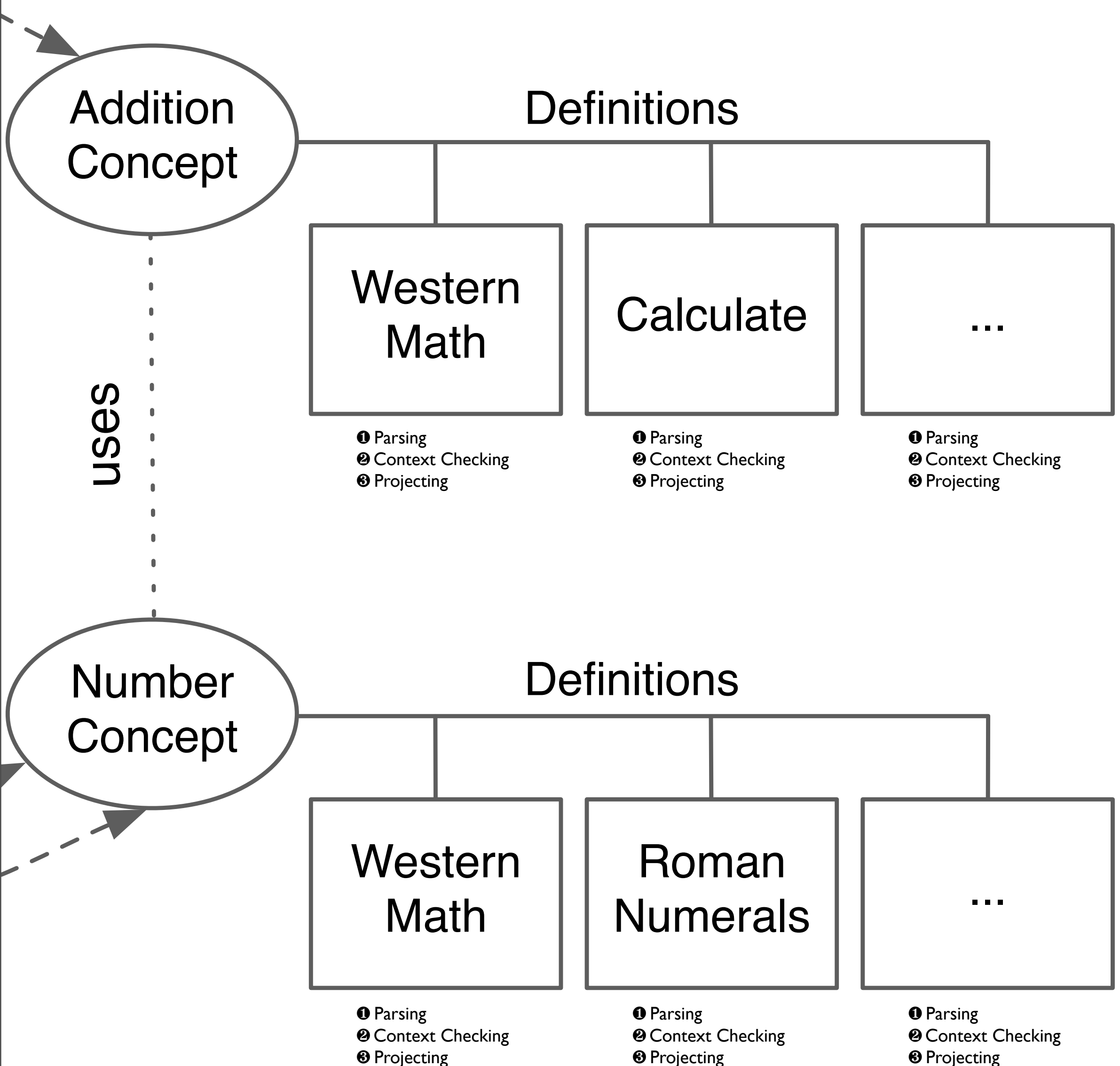
❶Parsing
❷Context Checking
❸Projecting



3+4

LANGUAGE DEFINITIONS

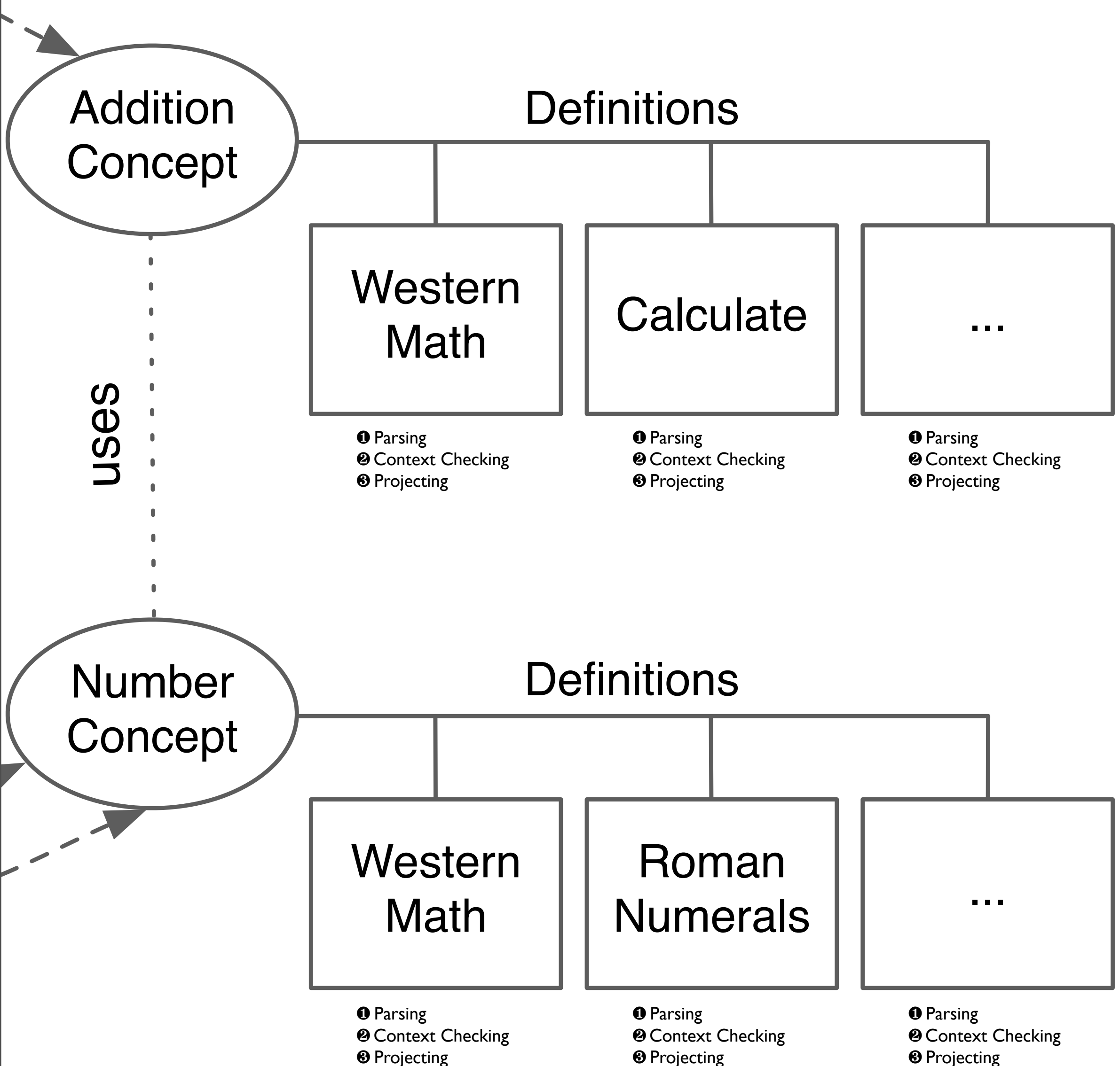
❶Parsing
❷Context Checking
❸Projecting



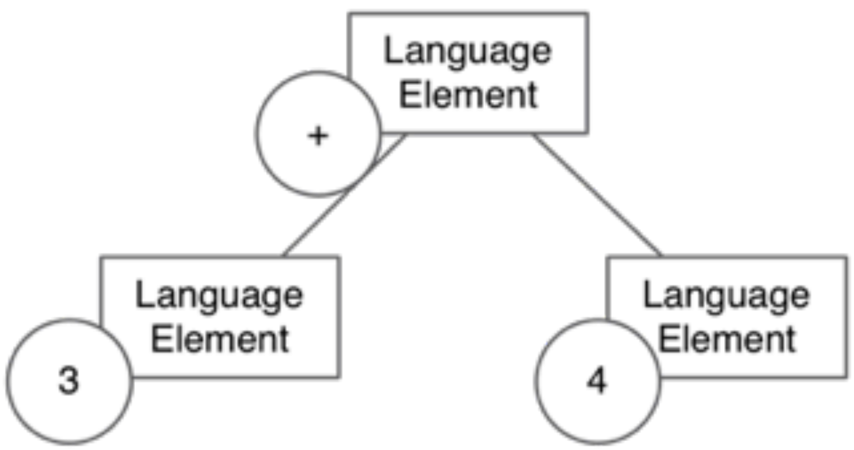
3+4 →

LANGUAGE DEFINITIONS

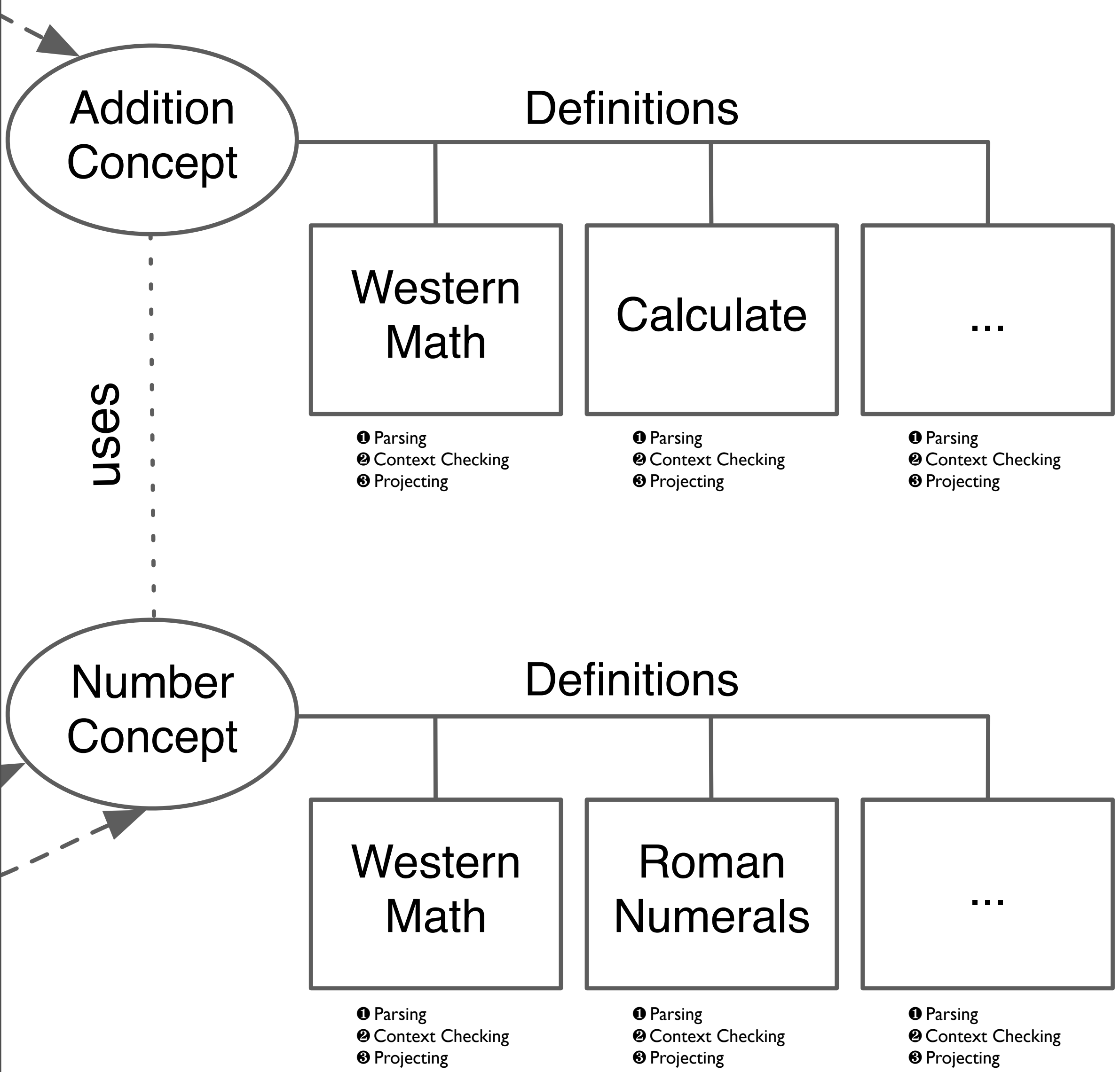
❶Parsing
❷Context Checking
❸Projecting



3+4 →



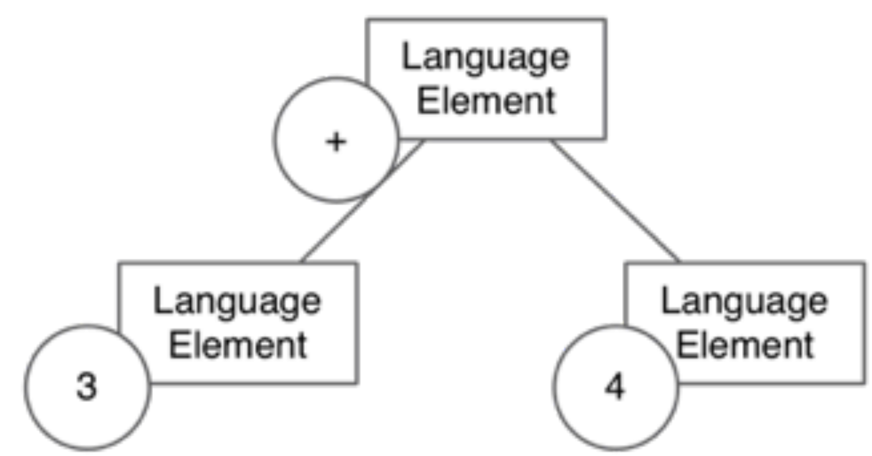
LANGUAGE DEFINITIONS



❶ Parsing
❷ Context Checking
❸ Projecting

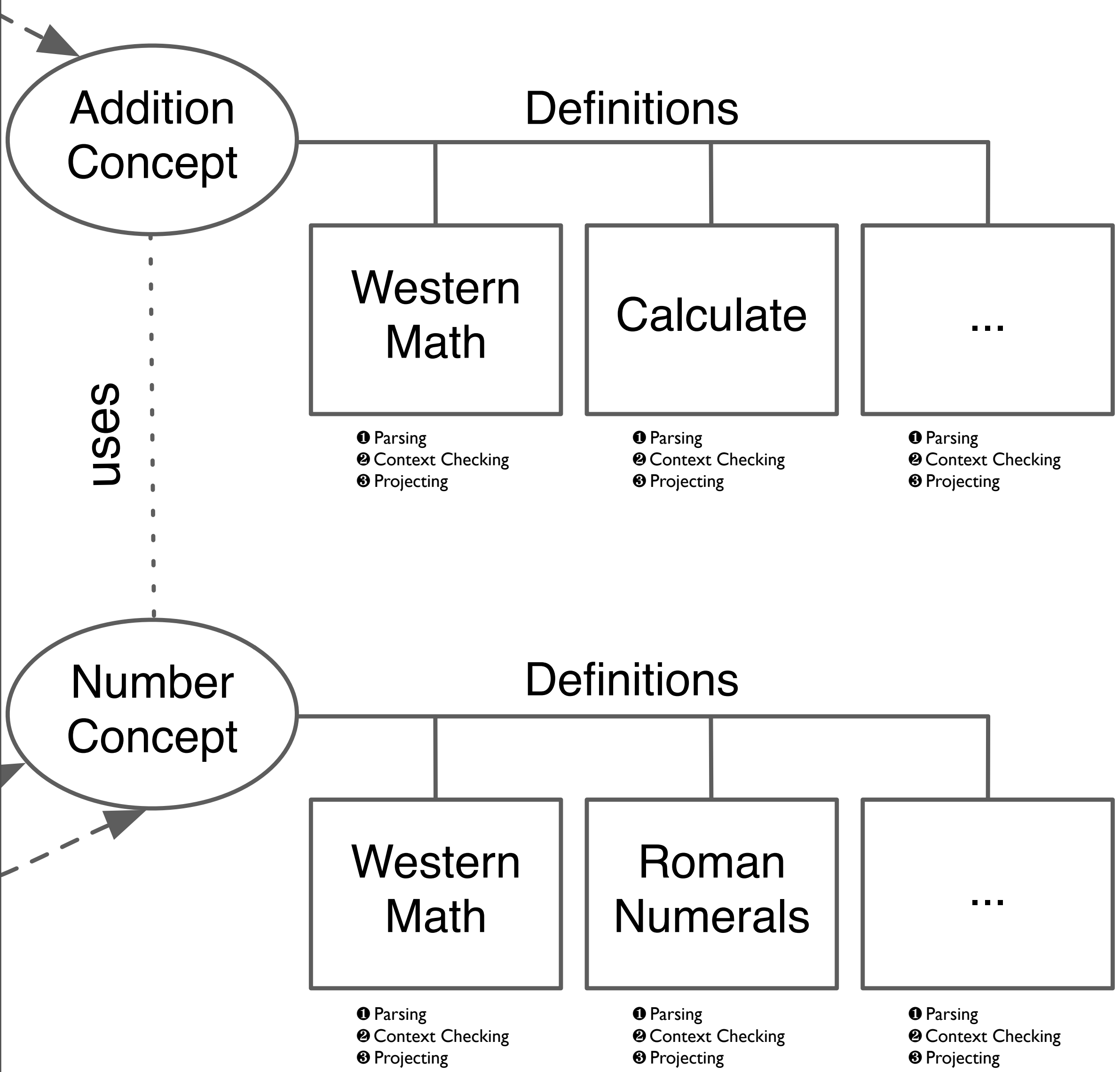
```
Addition ← term:t '+' factor:f {t,f}
```

3+4 →



LANGUAGE DEFINITIONS

LANGUAGE DEFINITIONS

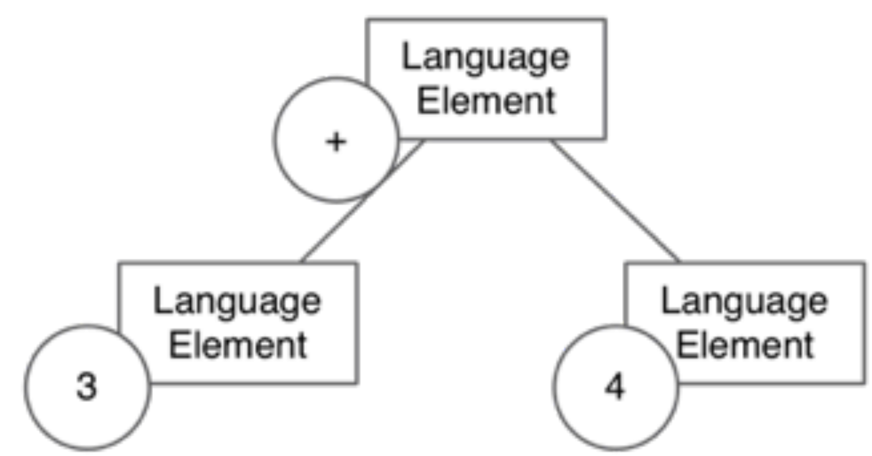


❶Parsing
❷Context Checking
❸Projecting

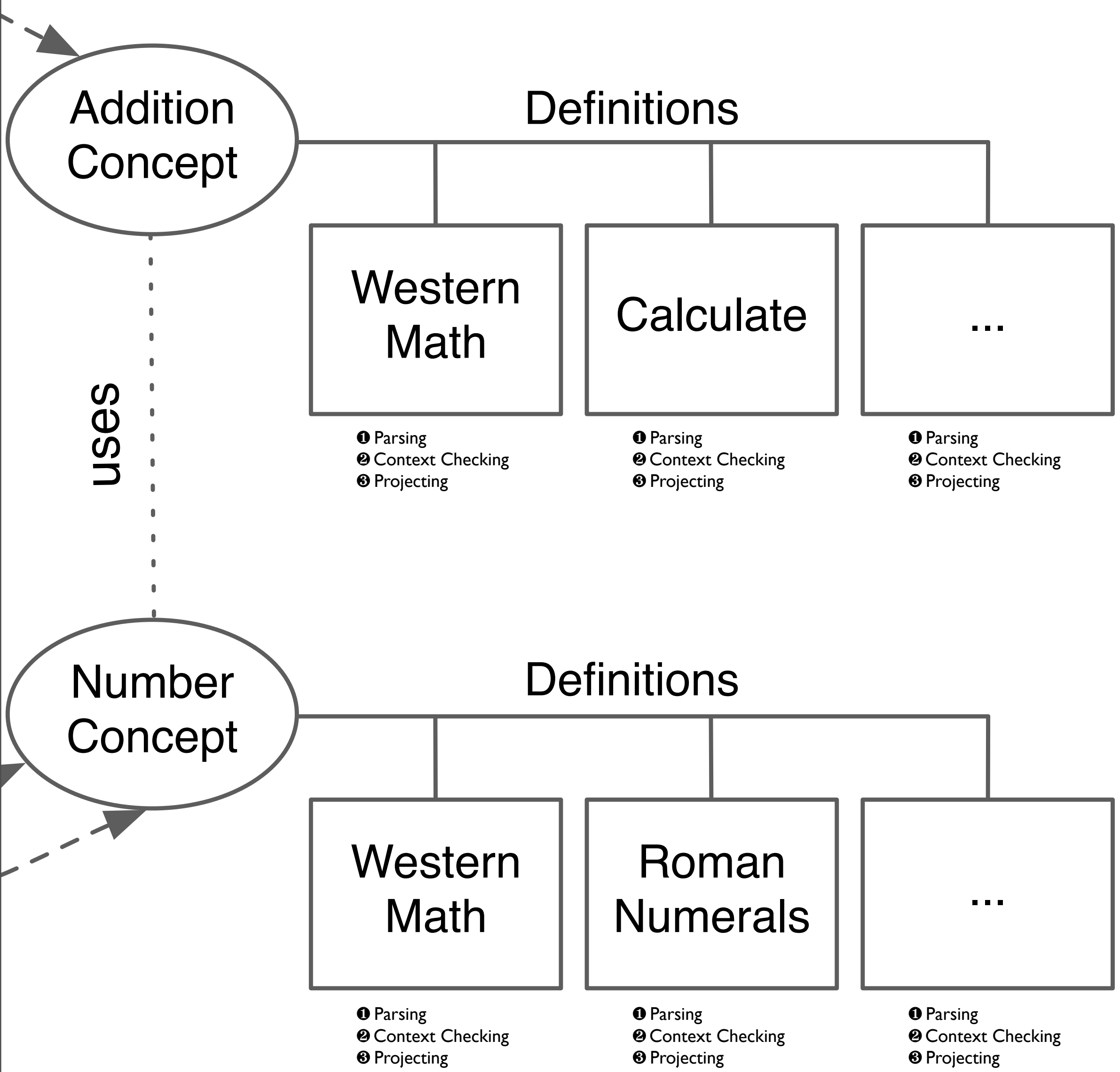
Addition ← term:t ‘+’ factor:f {t,f}

```
^self an: [t := self apply: #term]
n: [self a: $+]
n: [f := self apply: #factor]
n: [self le: {t. f}]
```

3+4 →

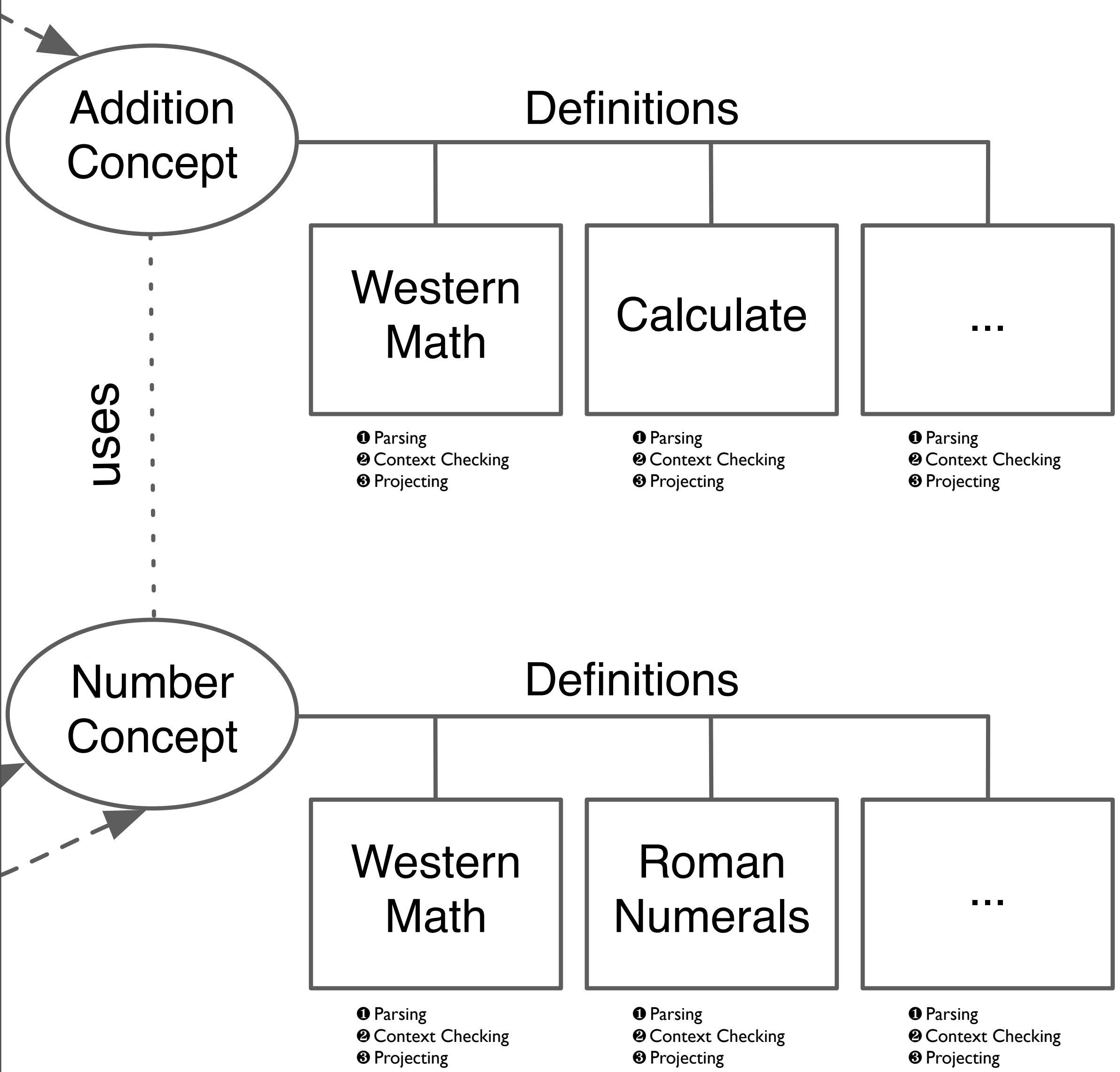


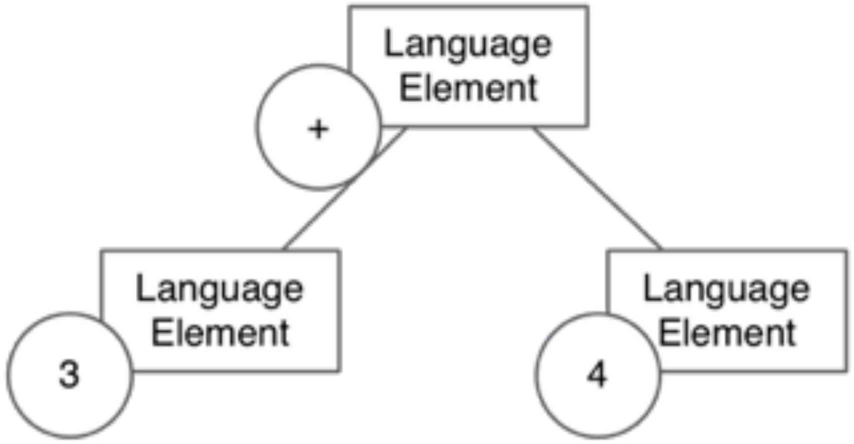
❶Parsing
❷Context Checking
❸Projecting



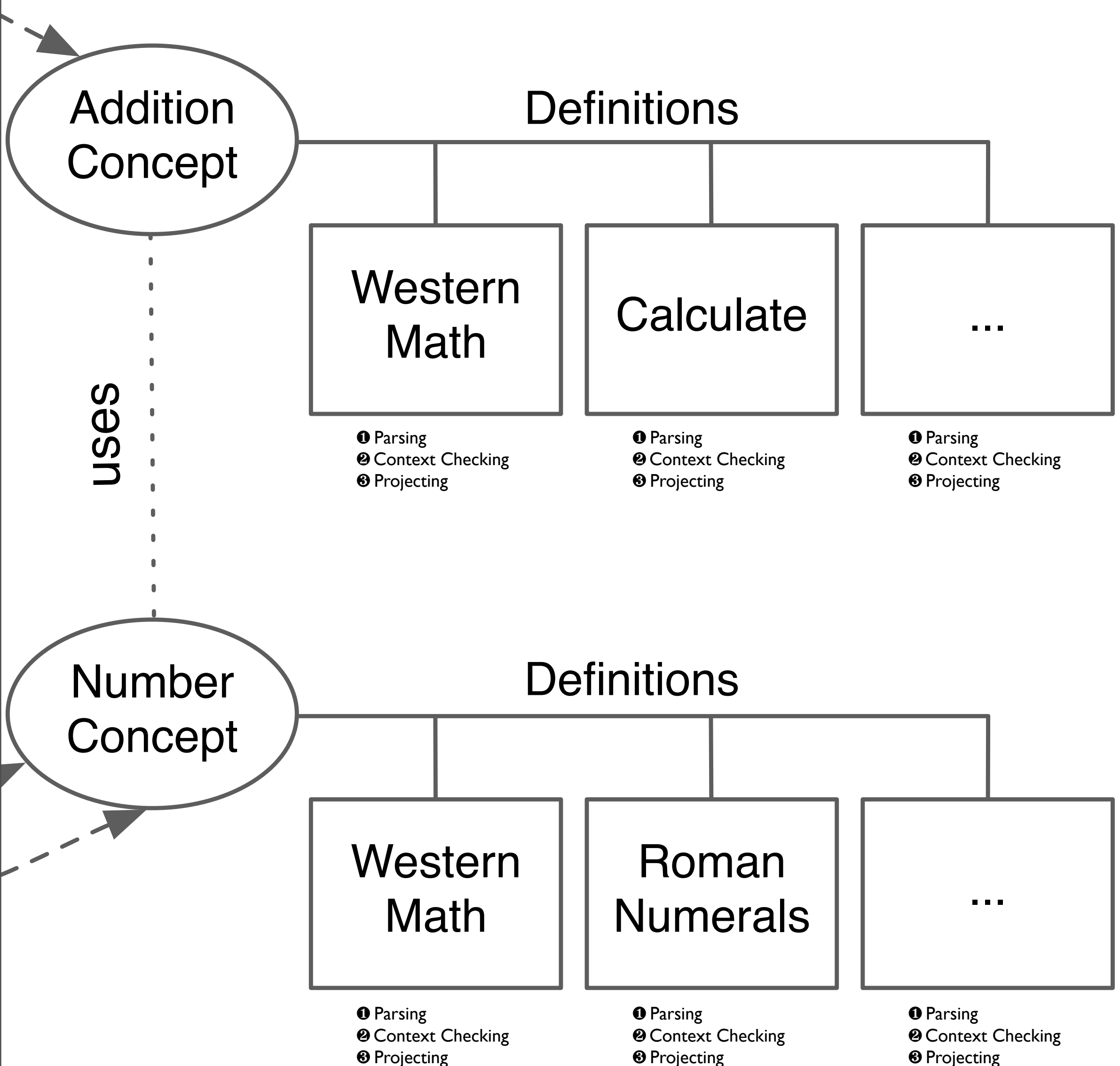
LANGUAGE DEFINITIONS

❶Parsing
❷Context Checking
❸Projecting

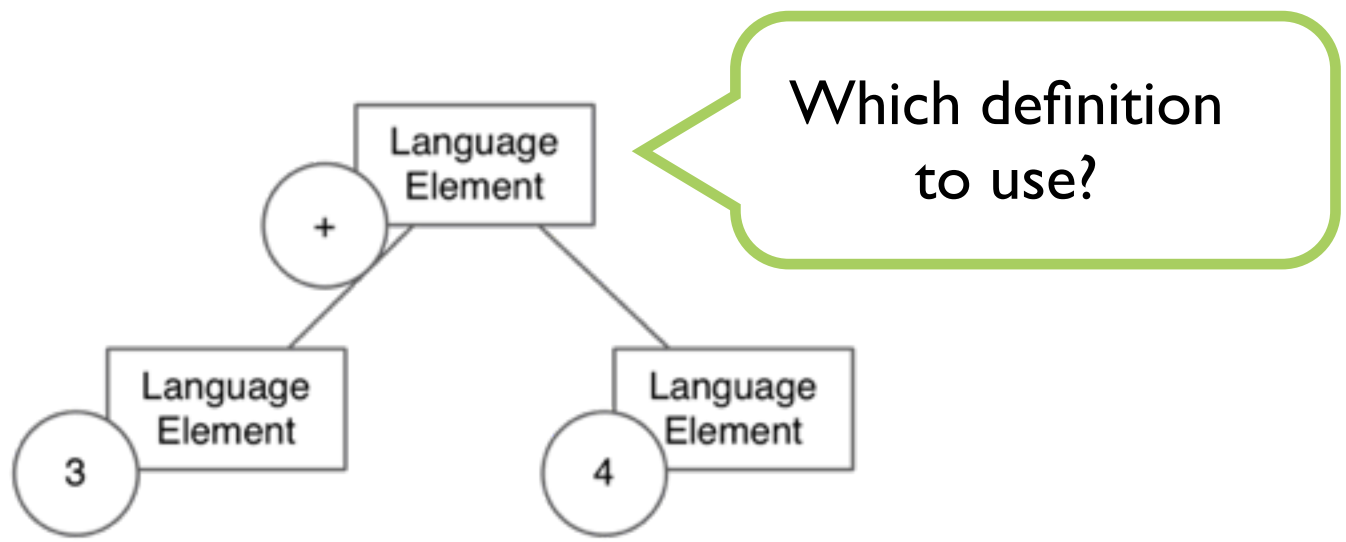




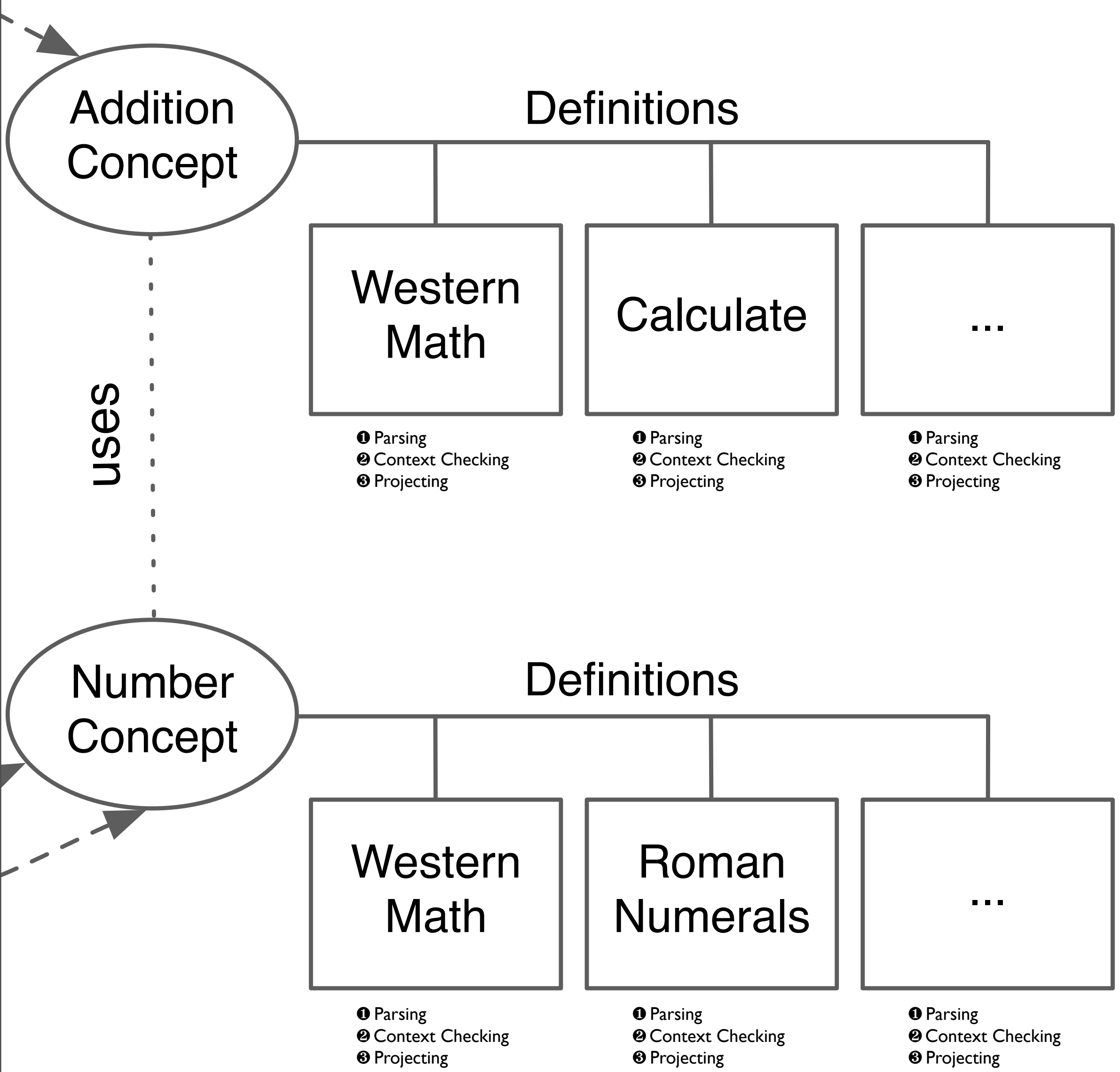
LANGUAGE DEFINITIONS



❶Parsing
❷Context Checking
❸Projecting



LANGUAGE DEFINITIONS

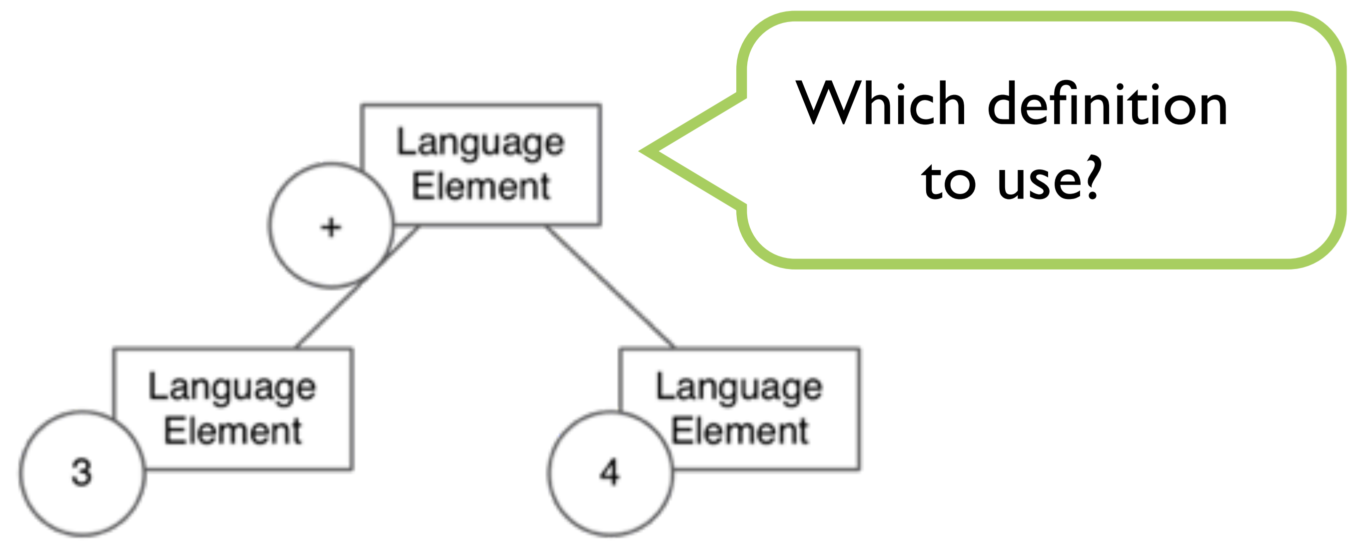


❶Parsing

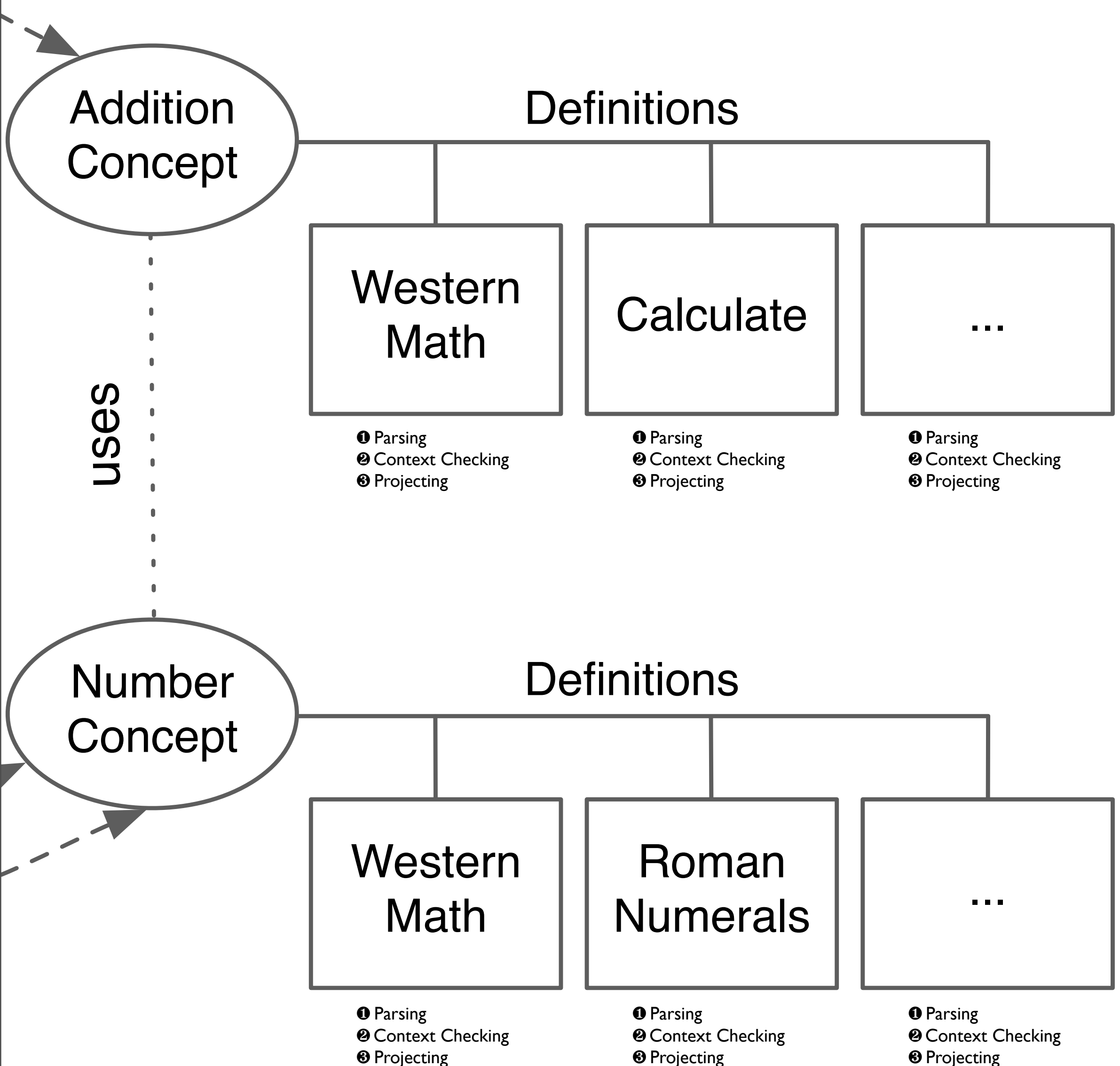
❷**Context Checking**

❸Projecting

```
^self language = 'Western Math'
```



LANGUAGE DEFINITIONS

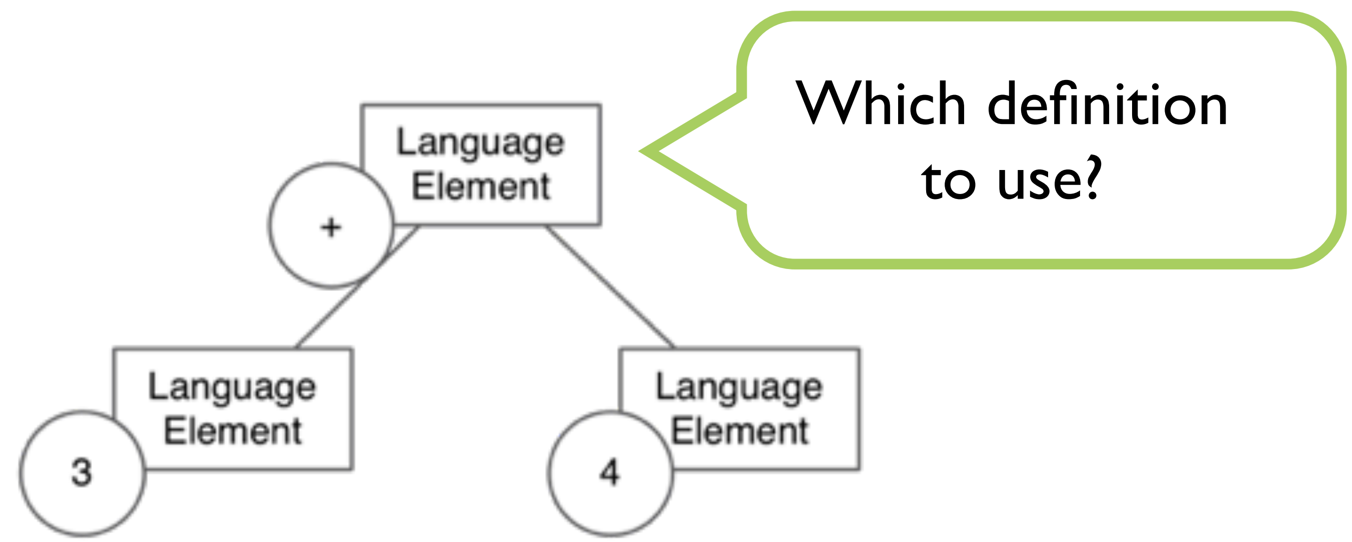


❶ Parsing

❷ Context Checking

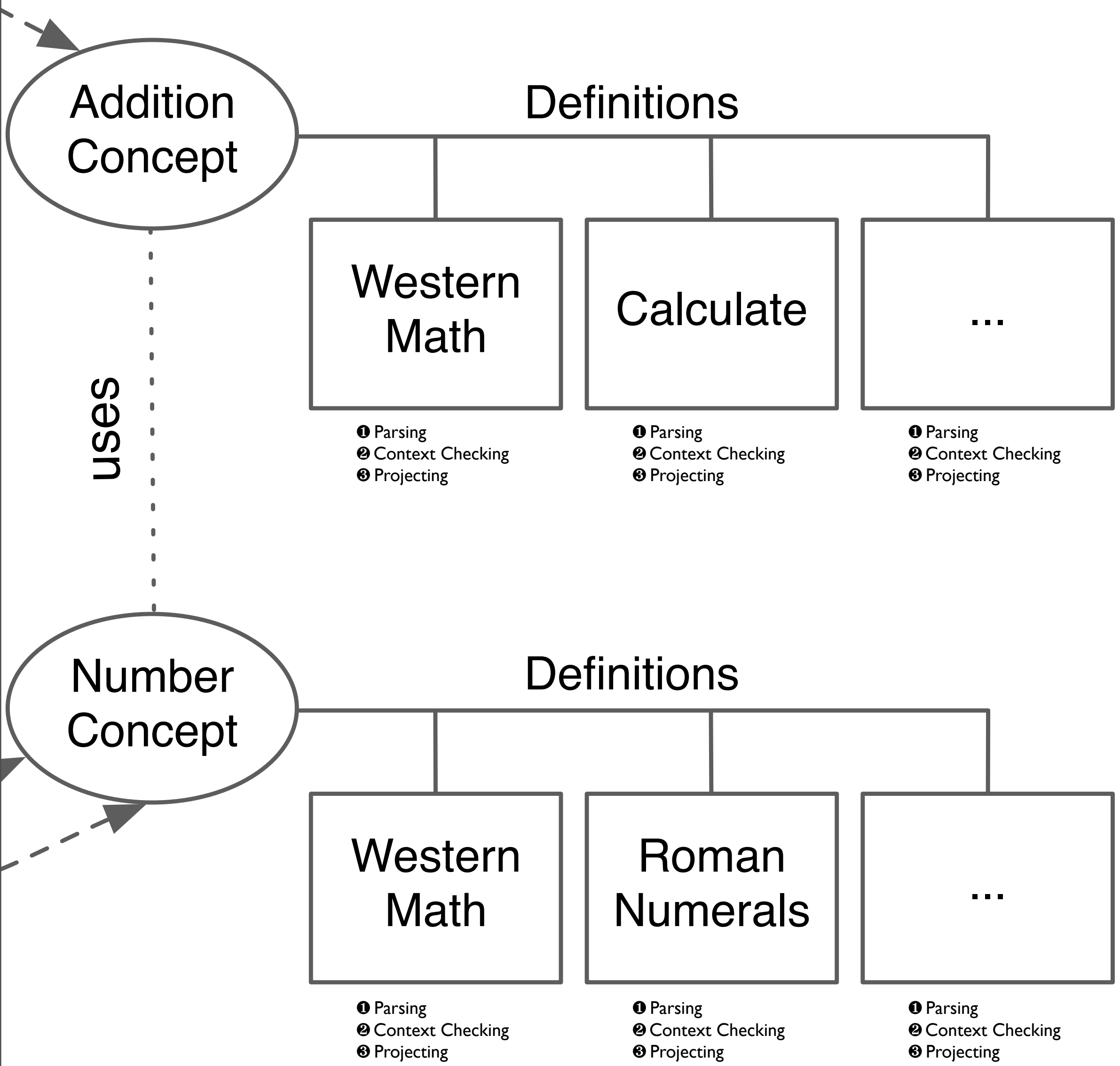
❸ Projecting

```
^self language = 'Western Math'
^self child1 value = 1
```

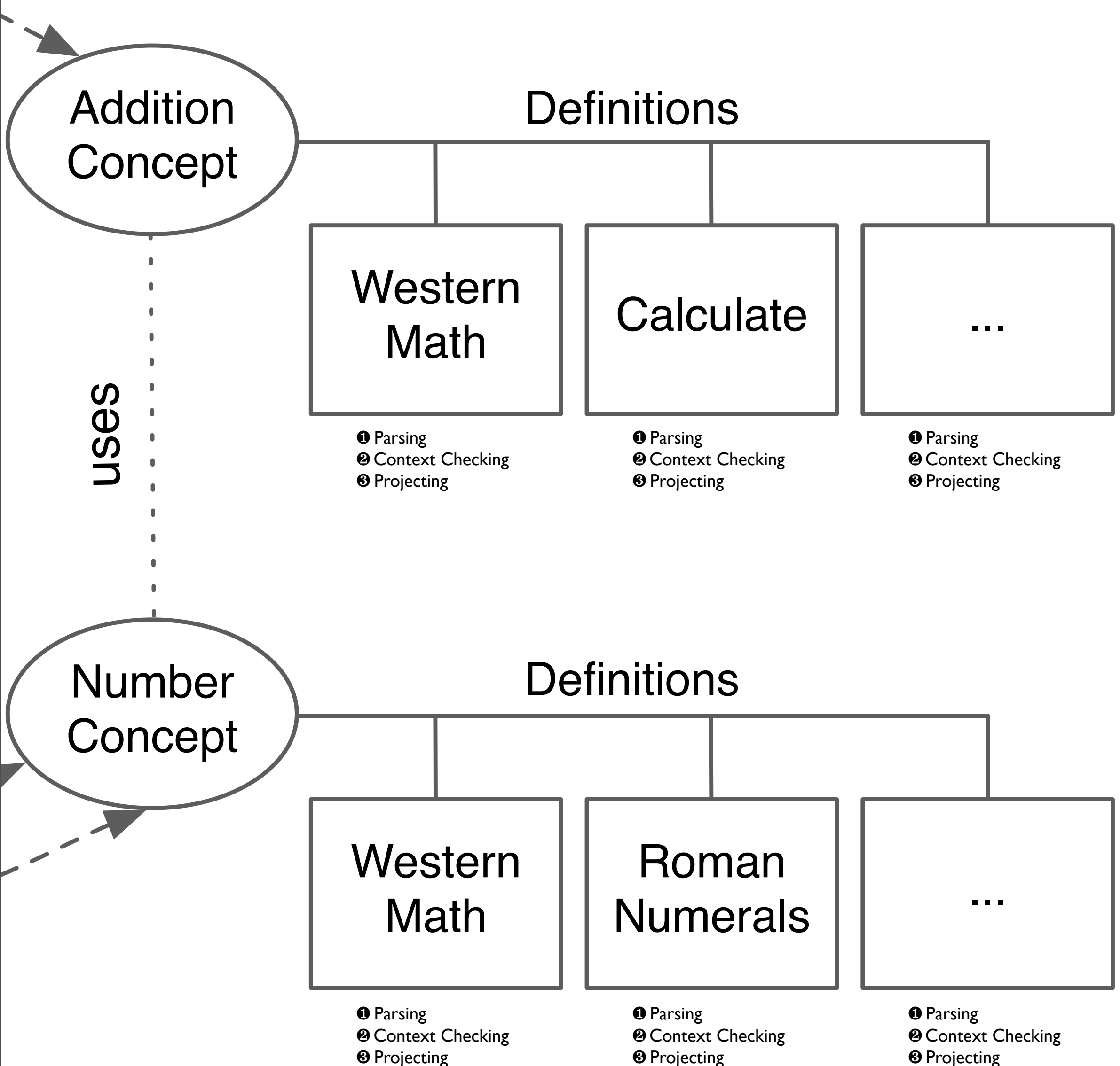


LANGUAGE DEFINITIONS

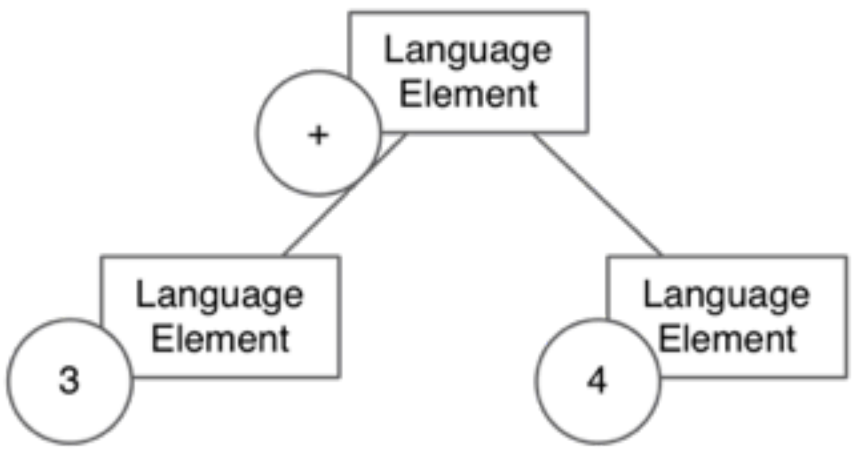
❶Parsing
❷Context Checking
❸Projecting



LANGUAGE DEFINITIONS

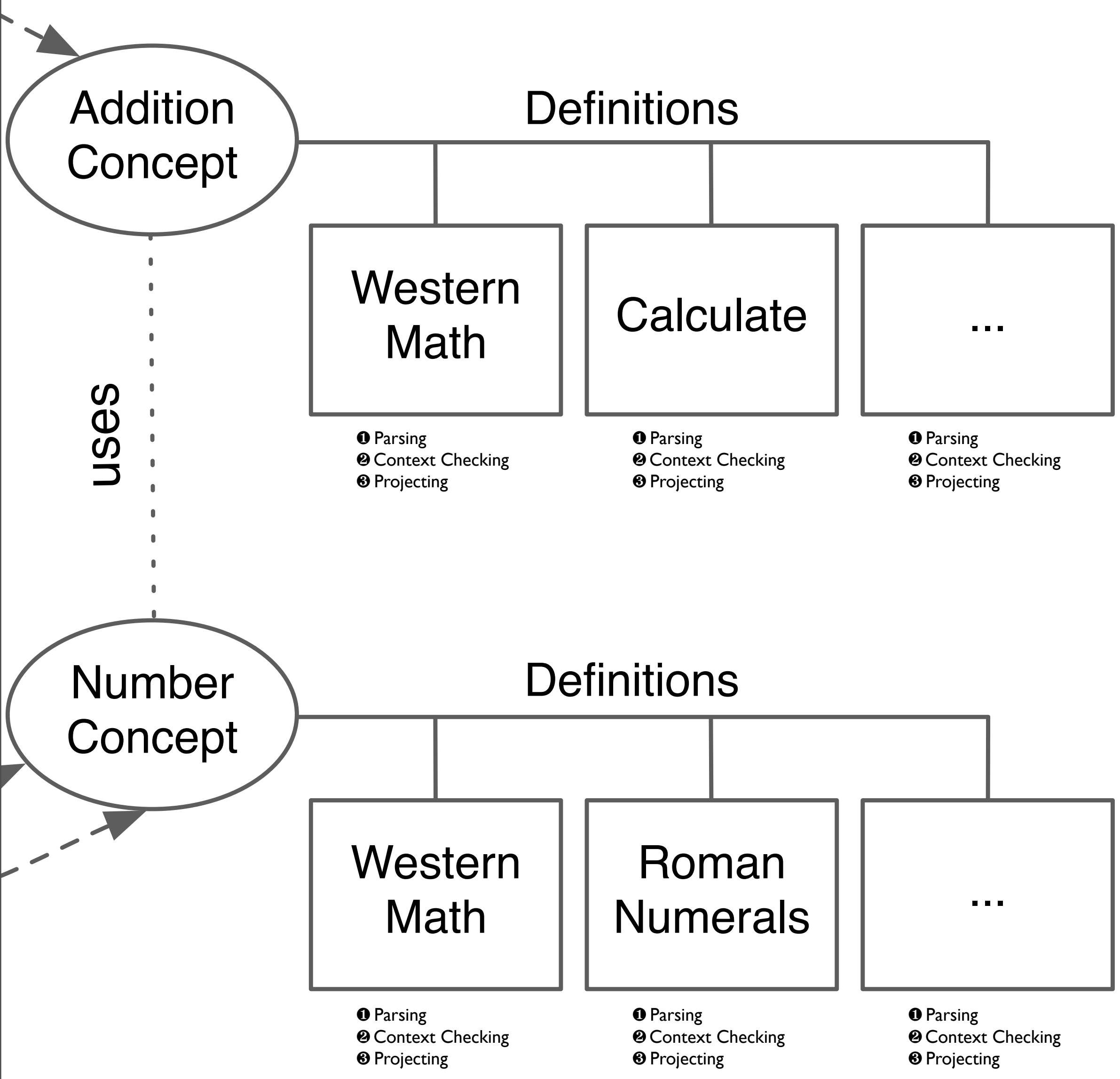


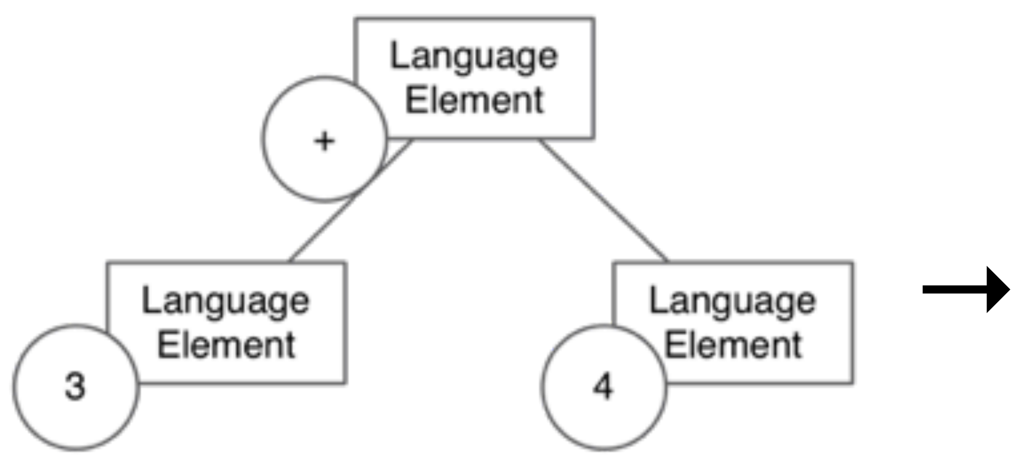
❶Parsing
❷Context Checking
❸Projecting



LANGUAGE DEFINITIONS

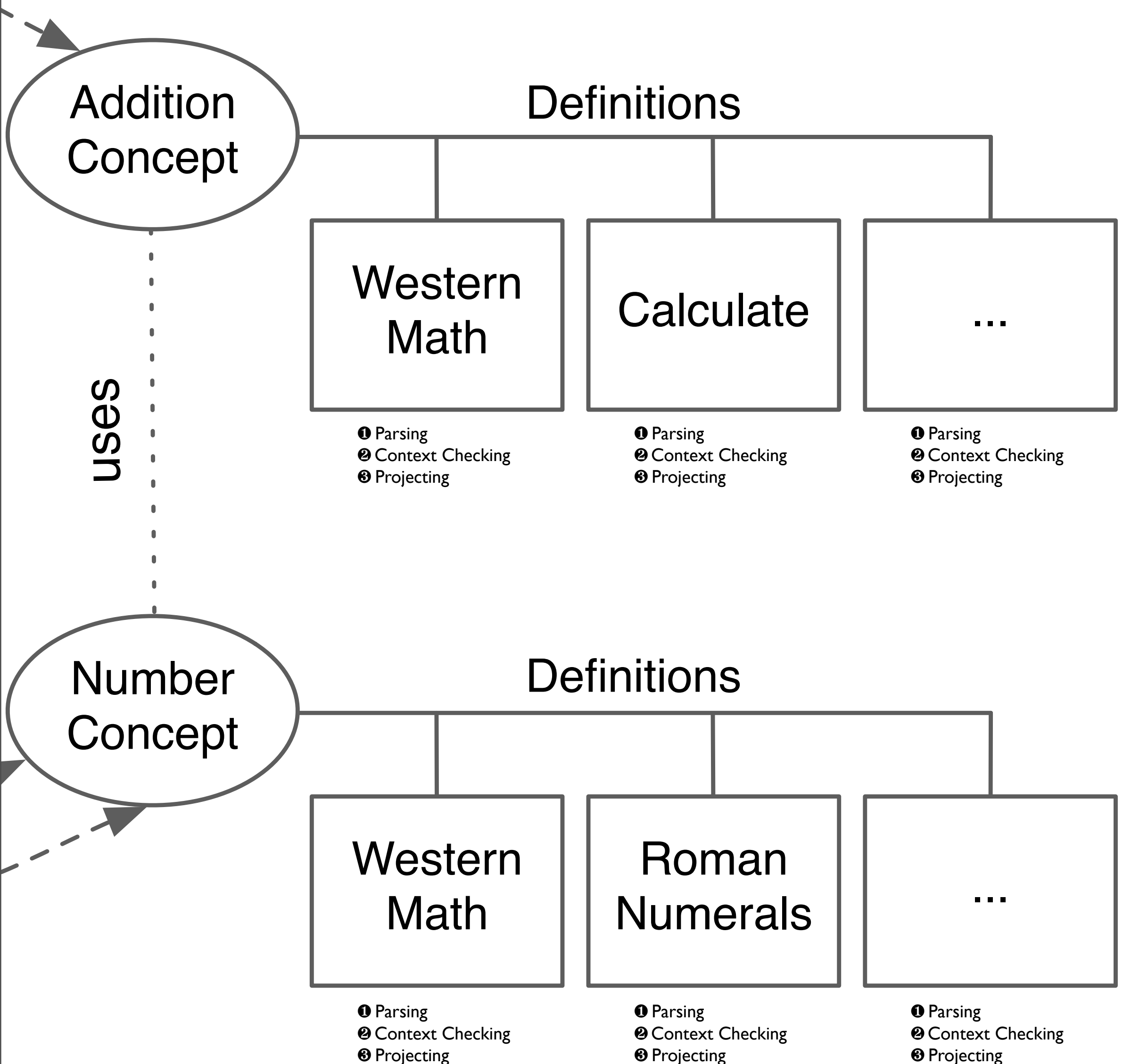
❶Parsing
❷Context Checking
❸Projecting

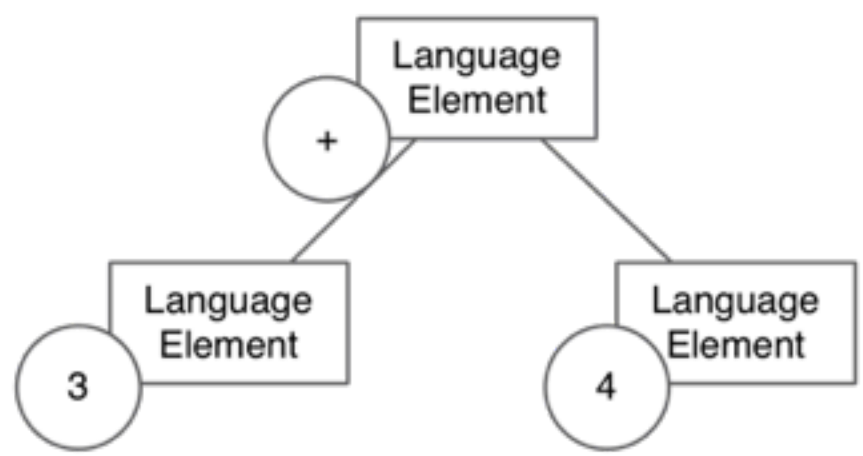




LANGUAGE DEFINITIONS

❶Parsing
❷Context Checking
❸Projecting





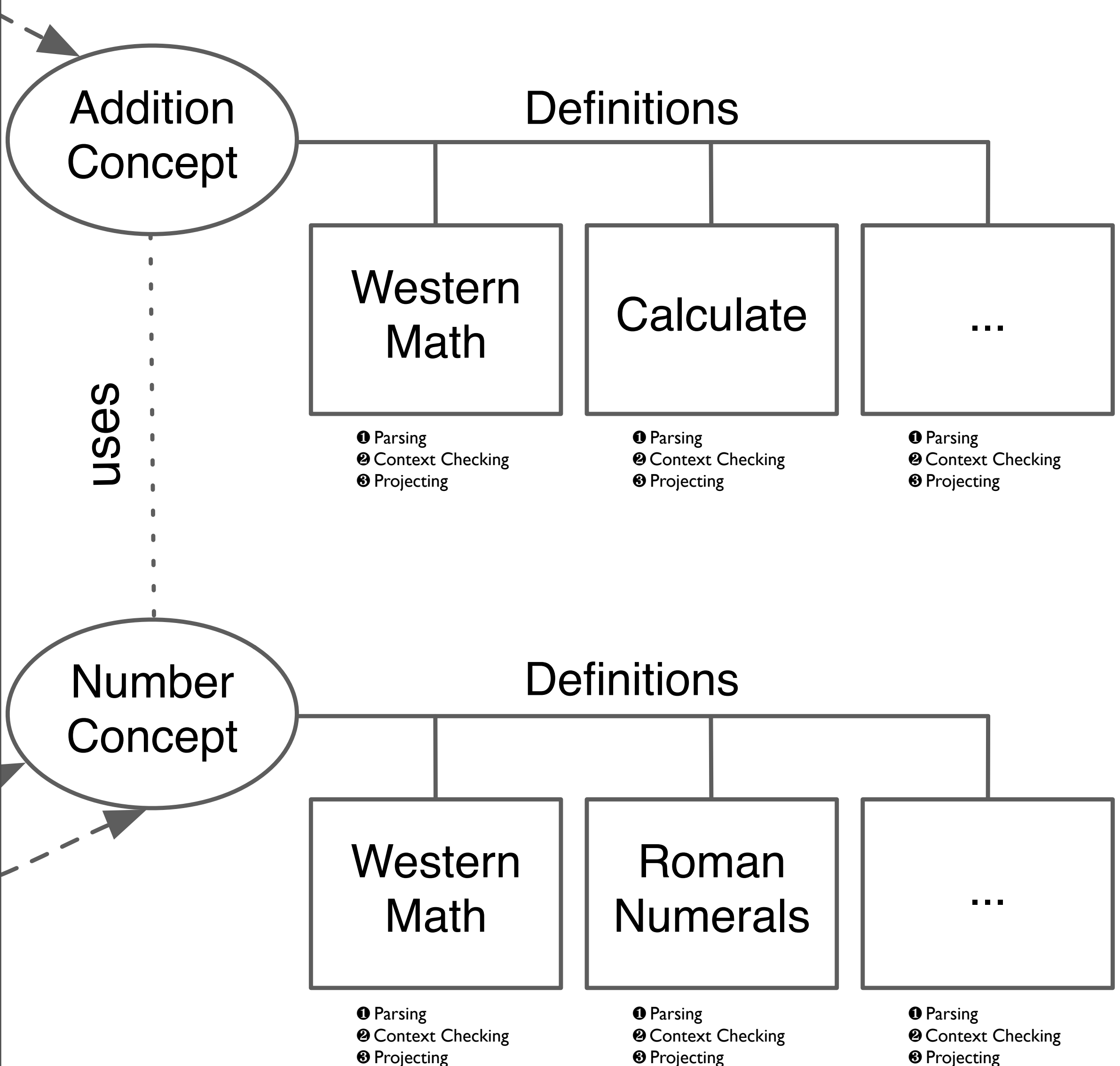
→ 3 + 4

LANGUAGE DEFINITIONS

❶Parsing

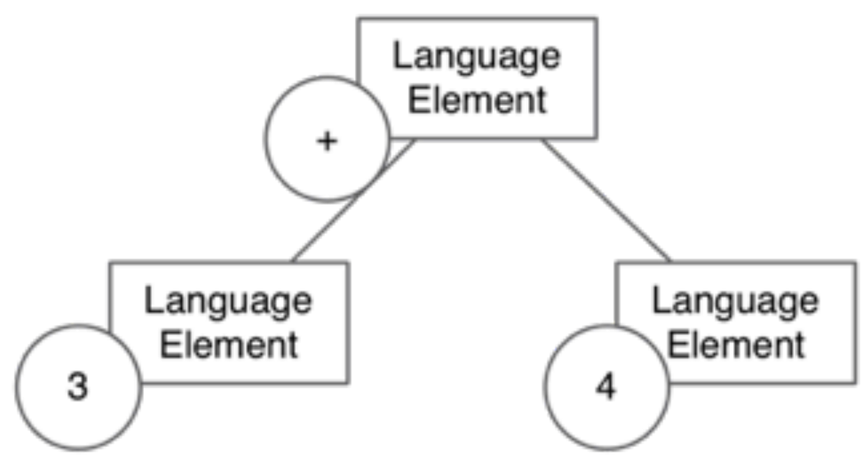
❷Context Checking

❸Projecting



Projectional pattern form

```
^{self lp1. $+ asText allBold. self lp2}
```



→ 3 **+** 4

LANGUAGE DEFINITIONS

QUICK TOUR

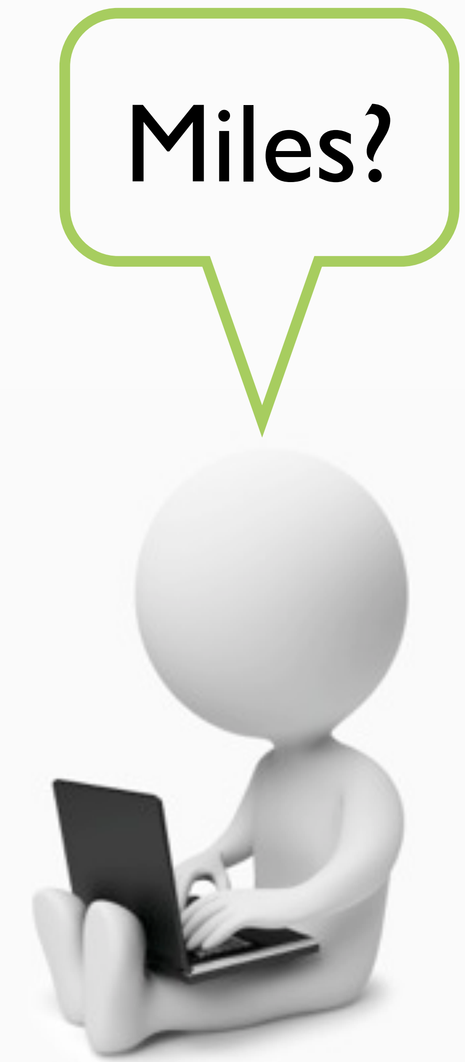
ACCOMMODATING CHANGE

ACCOMMODATING CHANGE

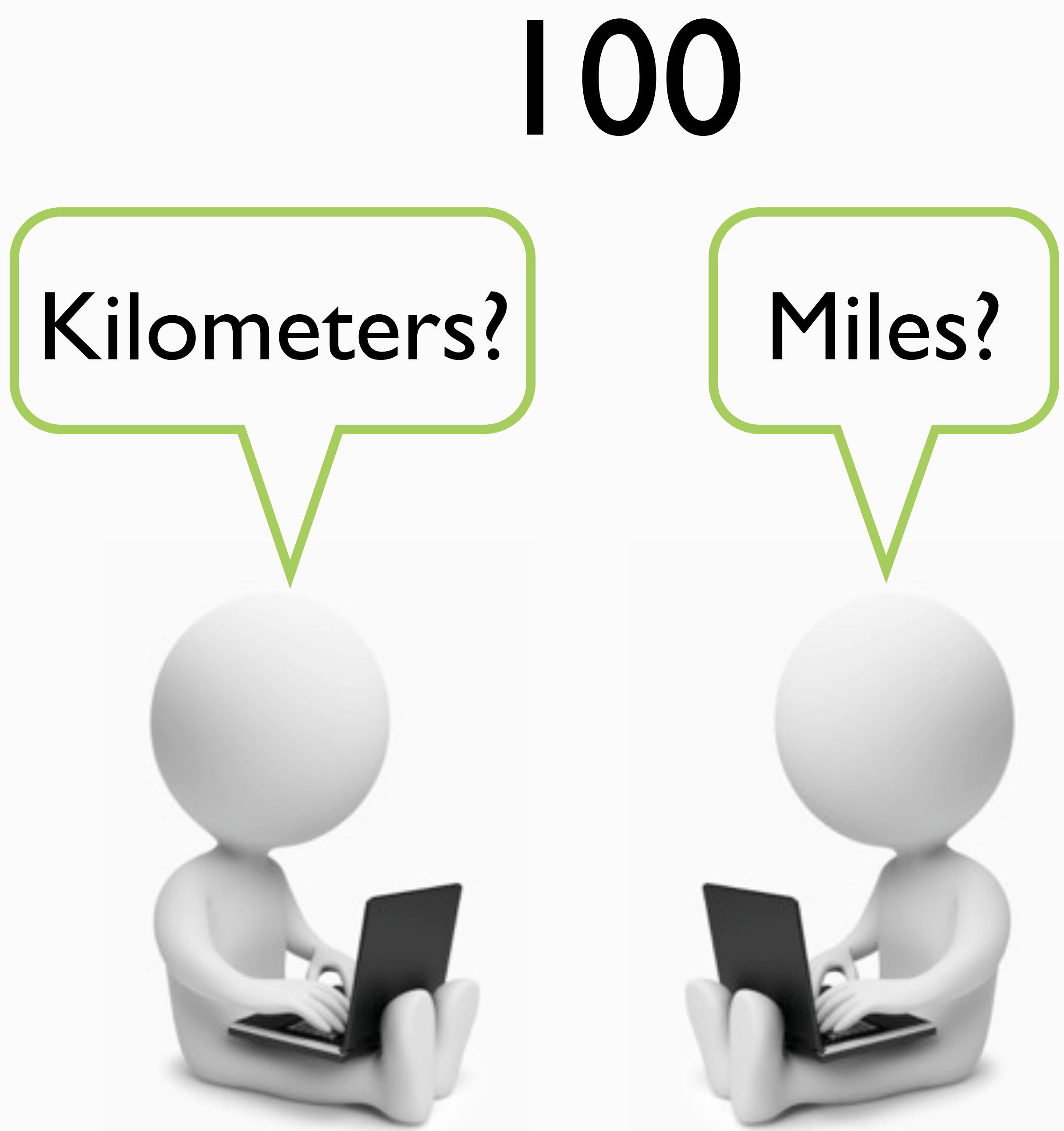
100

ACCOMMODATING CHANGE

100



ACCOMMODATING CHANGE



LANGUAGE OF LANGUAGES APPROACH

LANGUAGE OF LANGUAGES APPROACH

100

LANGUAGE OF LANGUAGES APPROACH

100

Number Concept

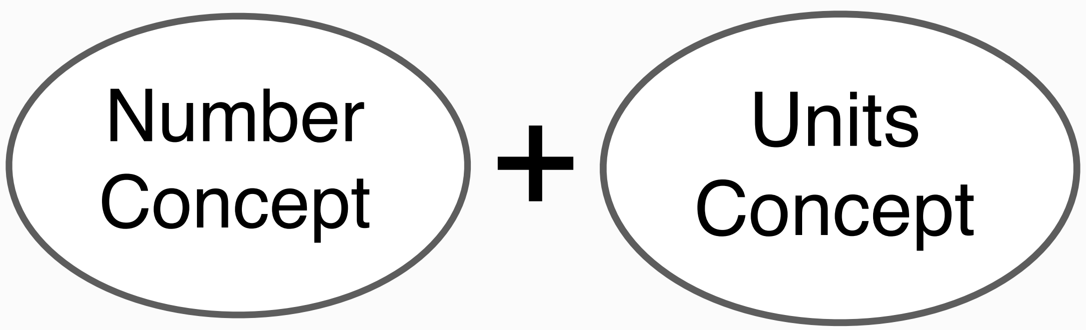
LANGUAGE OF LANGUAGES APPROACH

100

Number Concept +

LANGUAGE OF LANGUAGES APPROACH

100



UNITS DEMO

FUTURE WORK

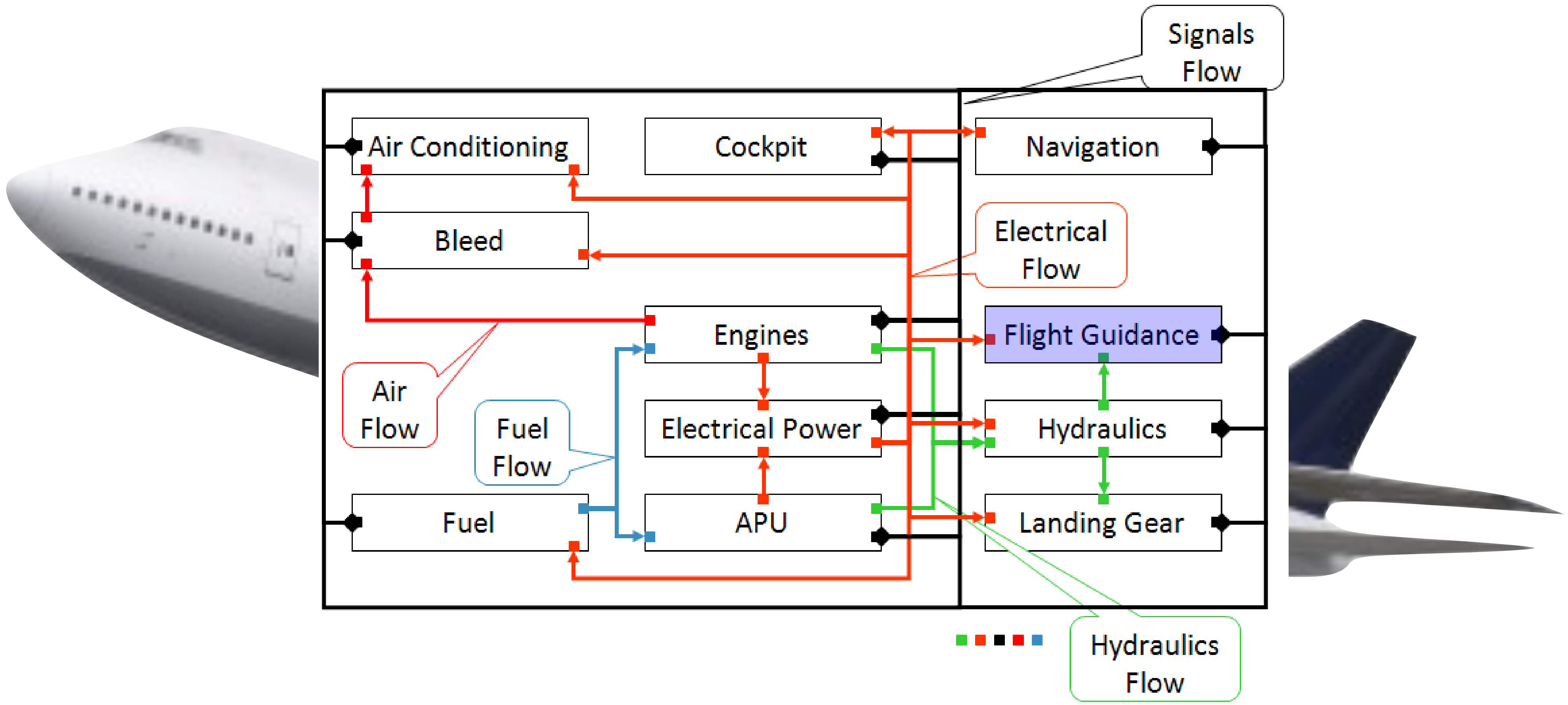
FUTURE WORK

Handle different notations within one domain

Models

Matlab	Modelica
Simulink	...

FUTURE WORK



Models

Matlab | Modelica
Simulink | ...

Architecture

AADL | SysML
UML | ...

Software

C/C++ | Fortran
Java | ...

FUTURE WORK

Models

Matlab	Modelica
Simulink	...

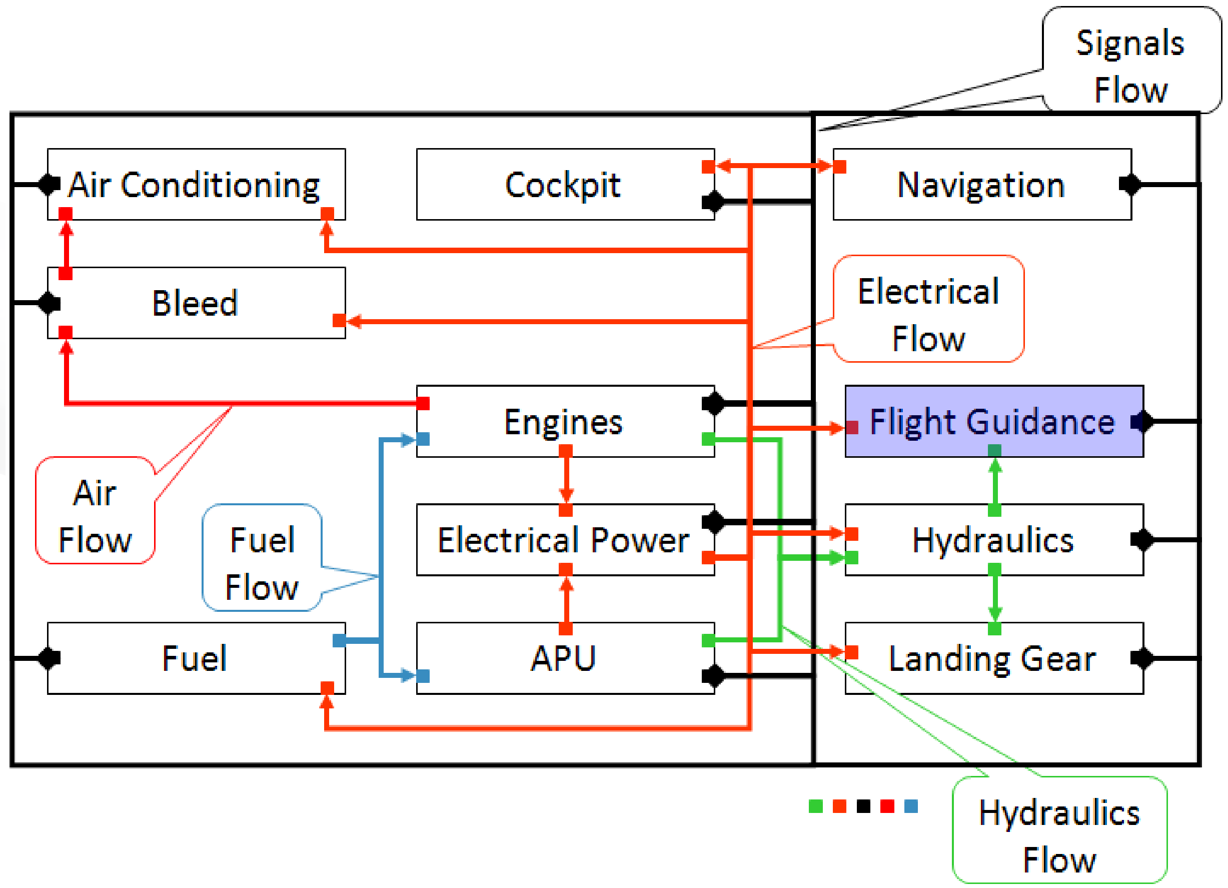
Architecture

AADL	SysML
UML	...

Software

C/C++	Fortran
Java	...

FUTURE WORK



Models

Matlab	Modelica
Simulink	...

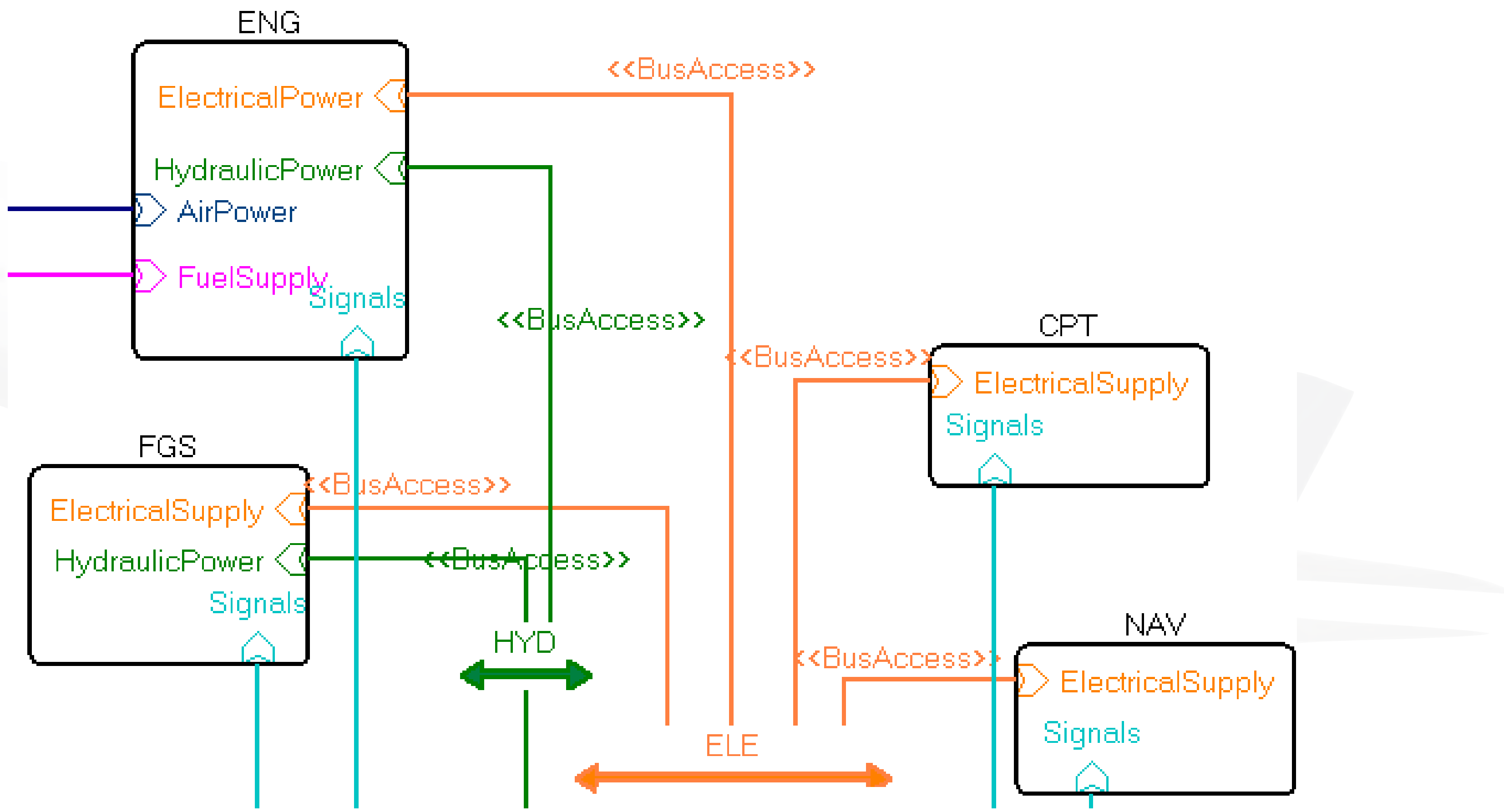
Architecture

AADL	SysML
UML	...

Software

C/C++	Fortran
Java	...

FUTURE WORK



Models

- Matlab
- Modelica
- Simulink
- ...

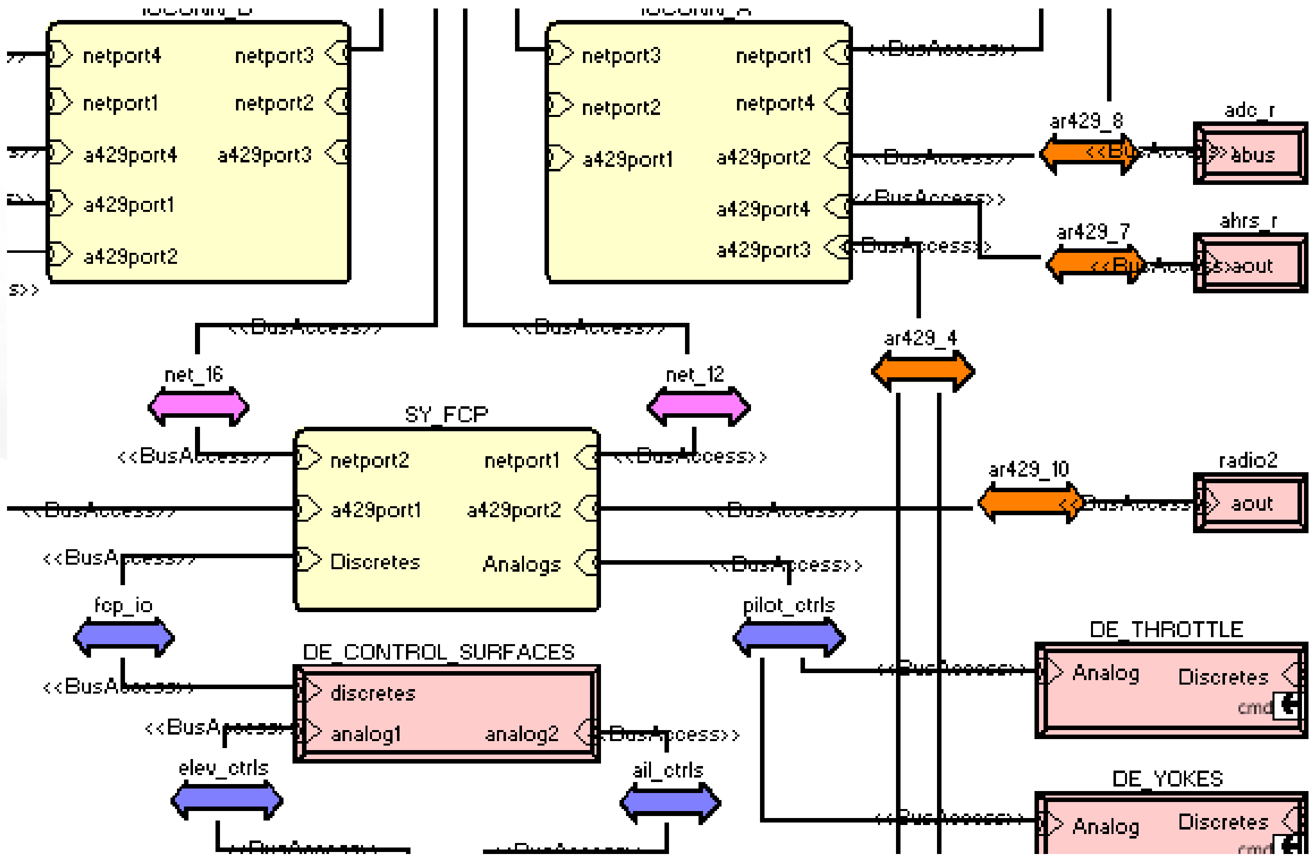
Architecture

- AADL
- SysML
- UML
- ...

Software

- C/C++
- Fortran
- Java
- ...

FUTURE WORK



Models

- Matlab
- Modelica
- Simulink
- ...

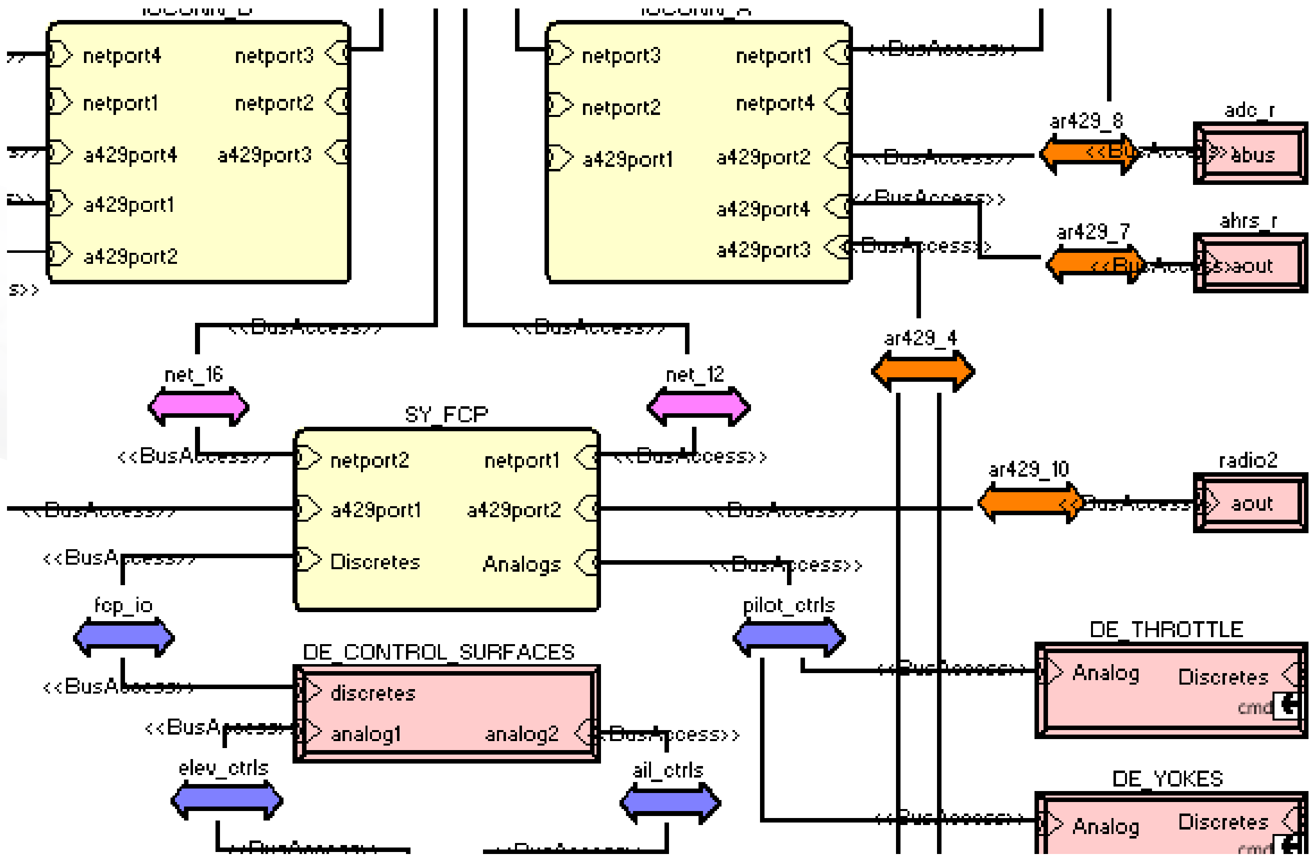
Architecture

- AADL
- SysML
- UML
- ...

Software

- C/C++
- Fortran
- Java
- ...

FUTURE WORK



Models

- Matlab
- Modelica
- Simulink
- ...

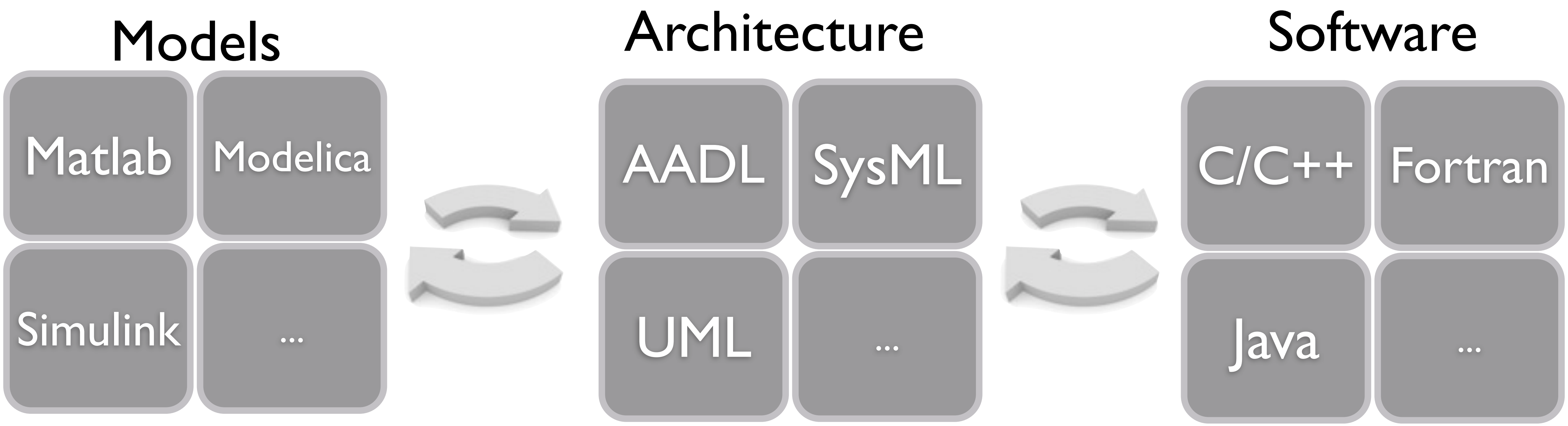
Architecture

- AADL
- SysML
- UML
- ...

Software

- C/C++
- Fortran
- Java
- ...

FUTURE WORK


Models
Matlab
Modelica
Simulink
...
Architecture
AADL
SysML
UML
...
Software
C/C++
Fortran
Java
...

FUTURE WORK

Models

Matlab	Modelica
Simulink	...

Architecture

AADL	SysML
UML	...

Software

C/C++	Fortran
Java	...

FUTURE WORK

Systems Architecture Virtual Integration

Models

- Matlab
- Modelica
- Simulink
- ...

Architecture

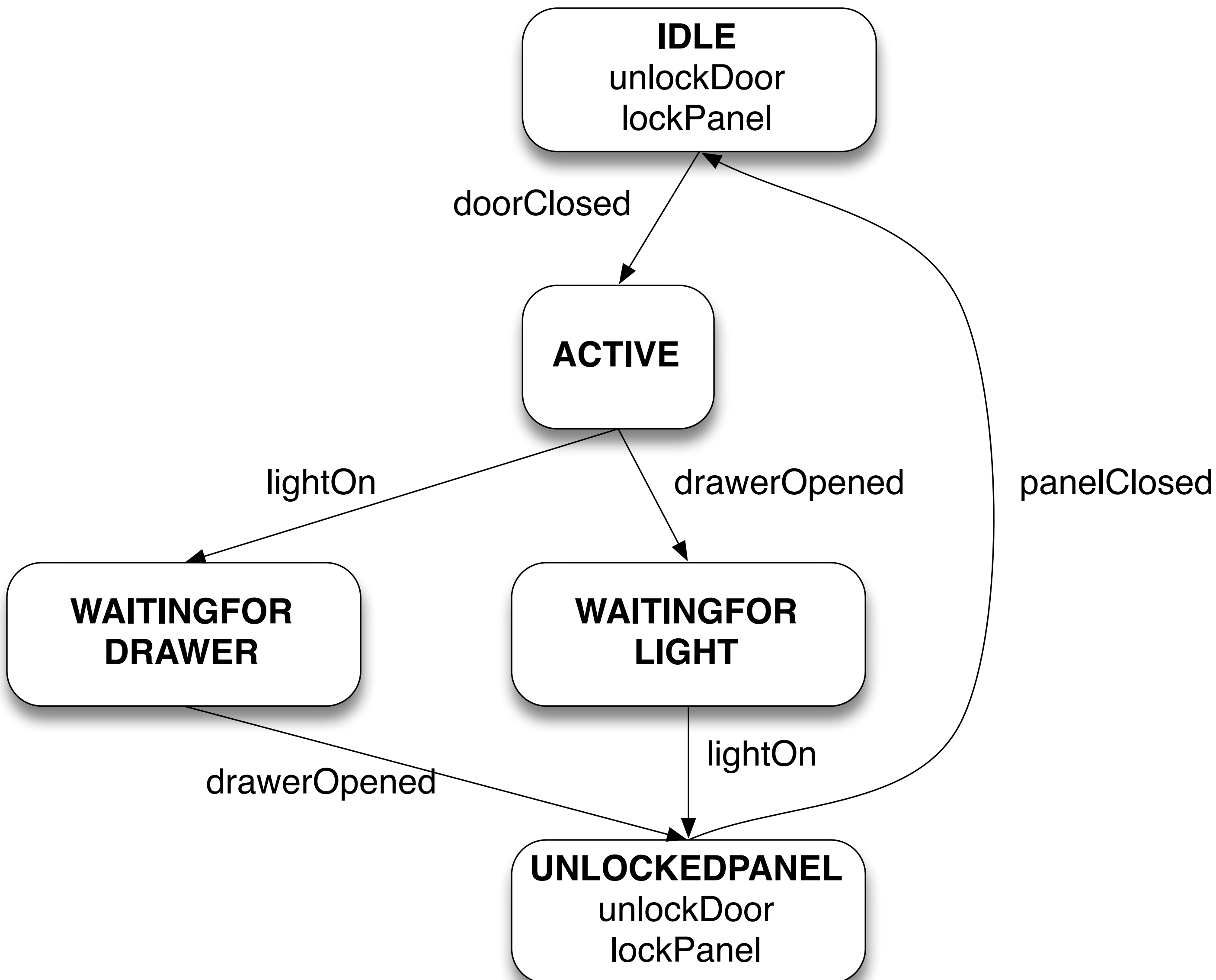
- AADL
- SysML
- UML
- ...

Software

- C/C++
- Fortran
- Java
- ...

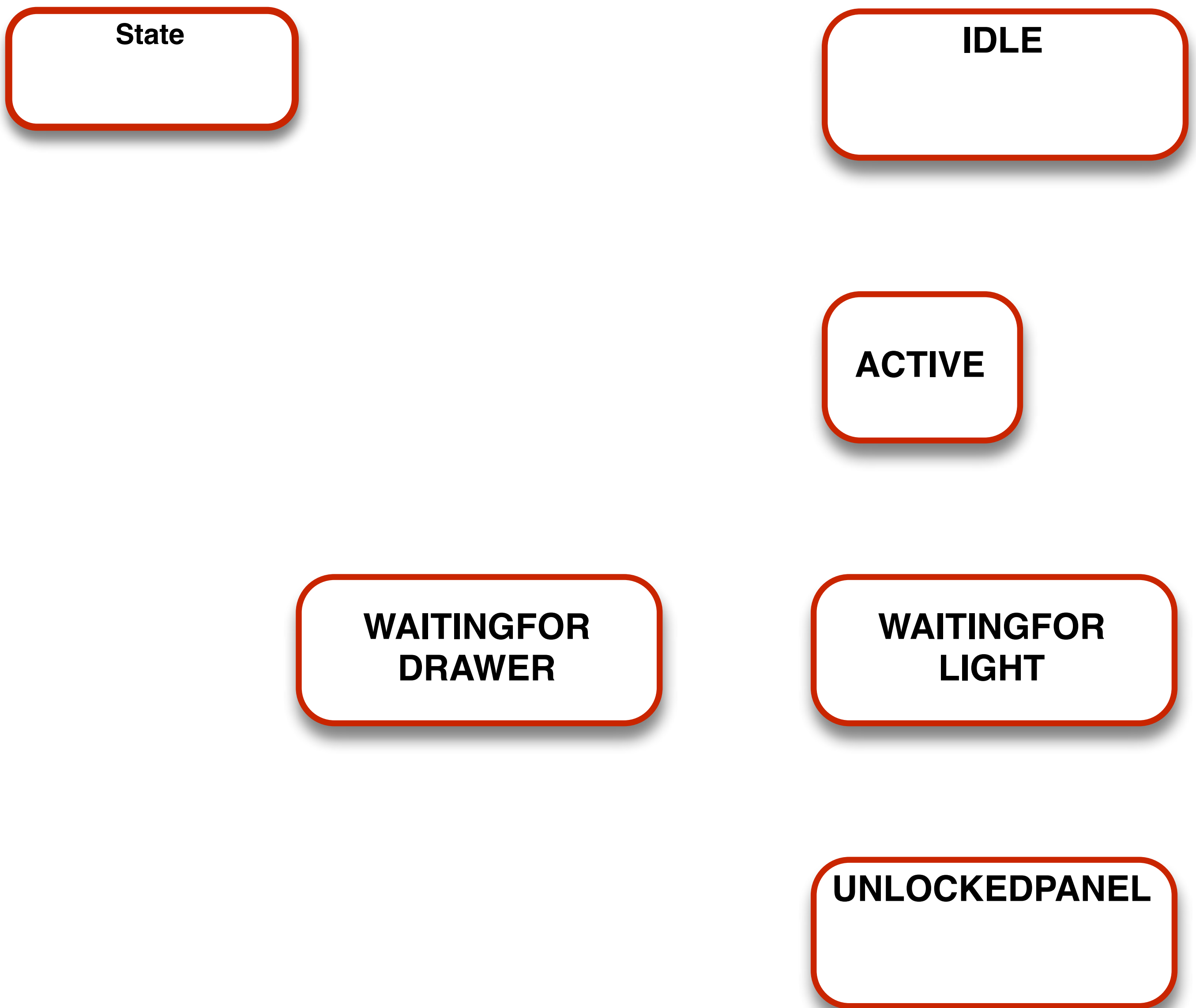
STATE MACHINE EXAMPLE

STATE MACHINE EXAMPLE



DECOMPOSE INTO CONCEPTS

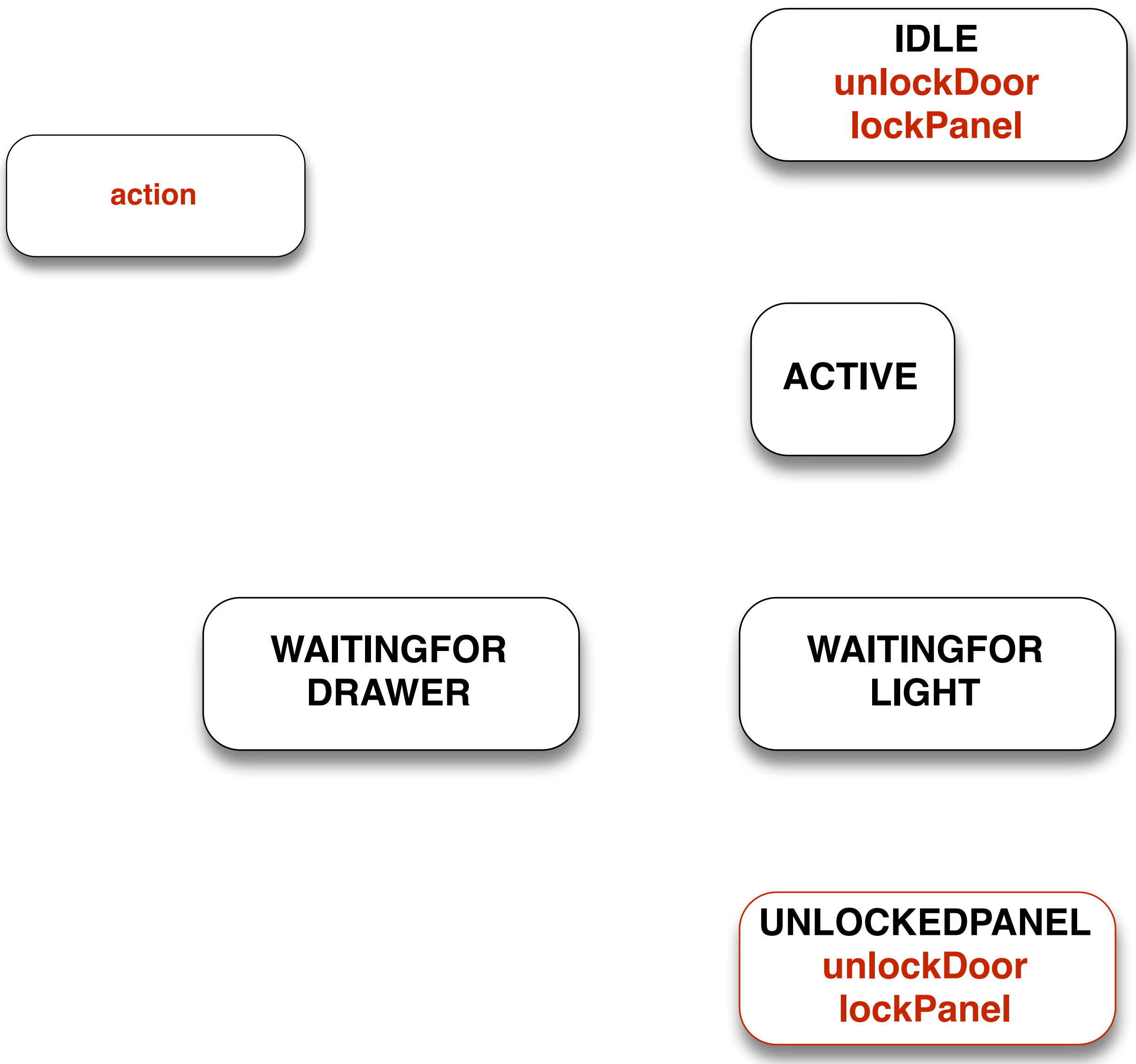
❶ State



DECOMPOSE INTO CONCEPTS

❶ State

❷ Actions



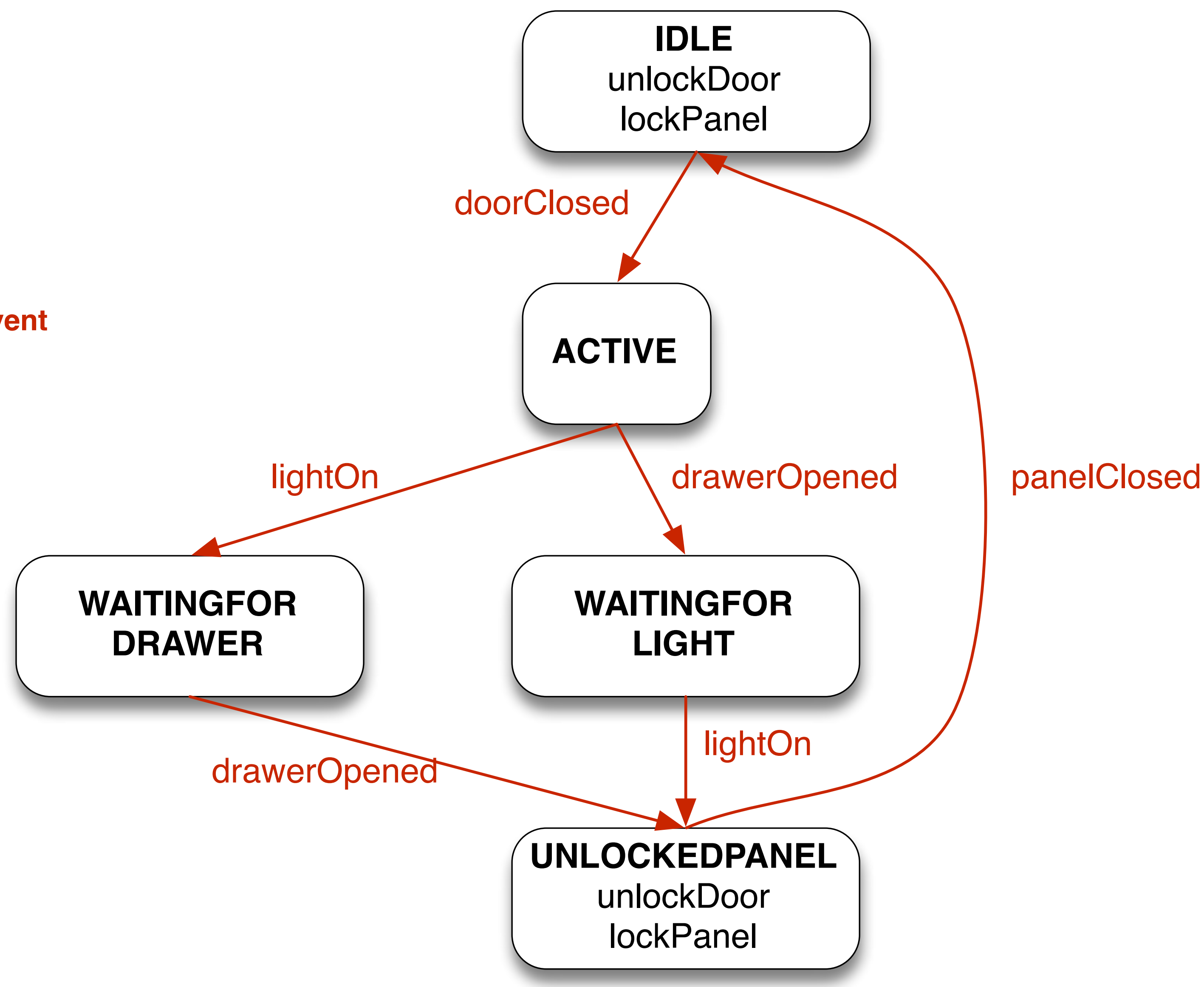
DECOMPOSE INTO CONCEPTS

❶ State

❷ Actions

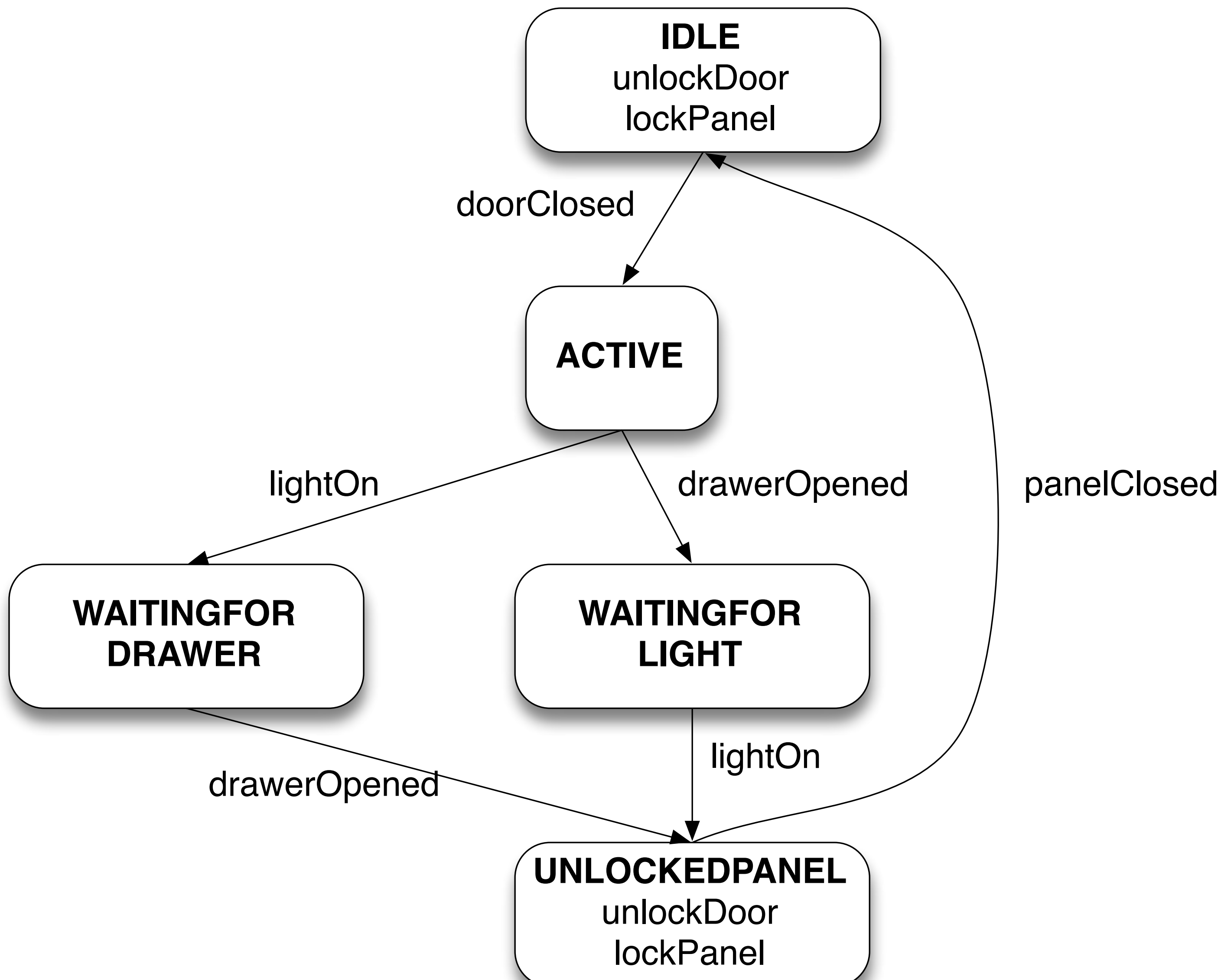
❸ Transitions

event

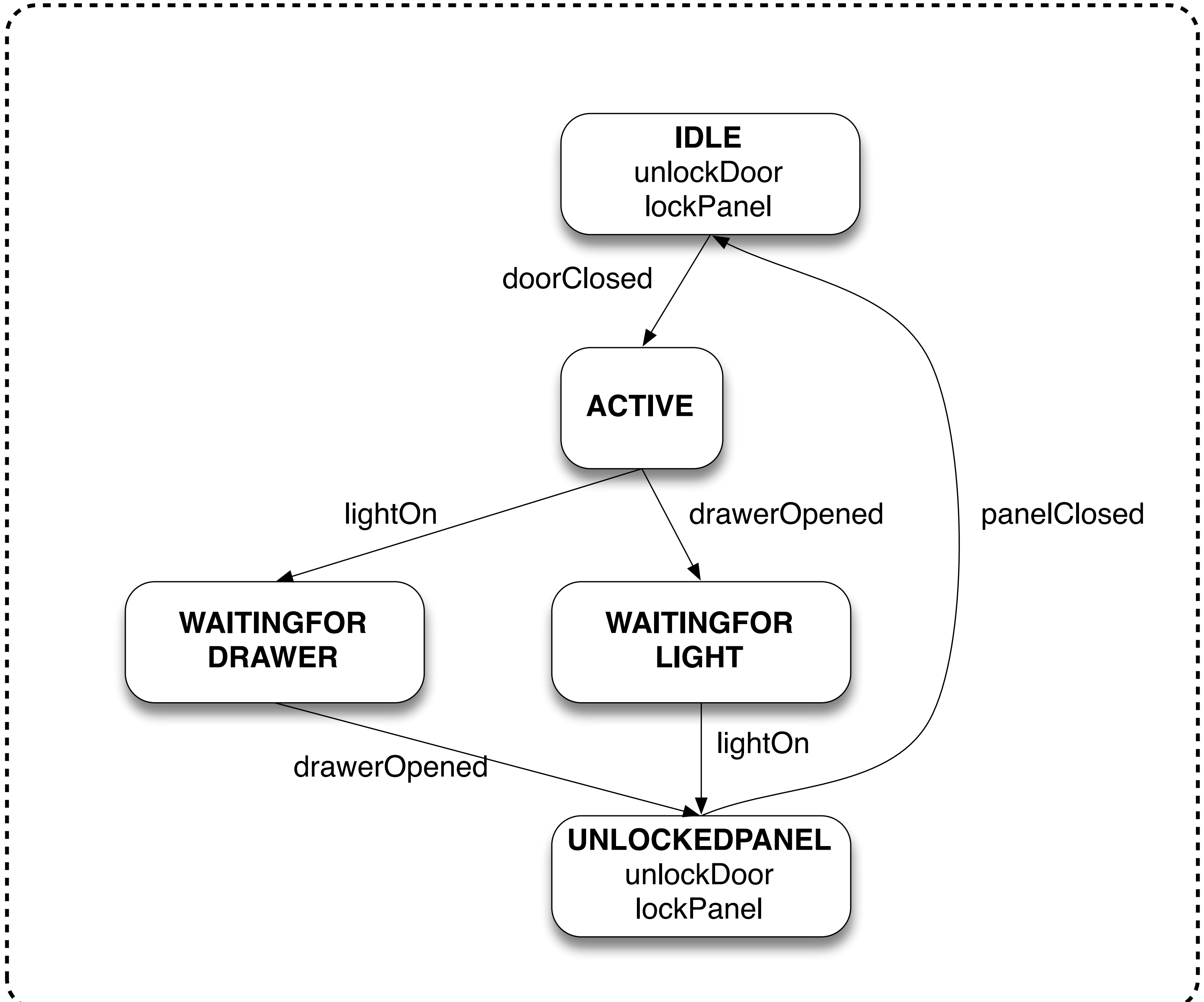


UNIFY CONCEPTS

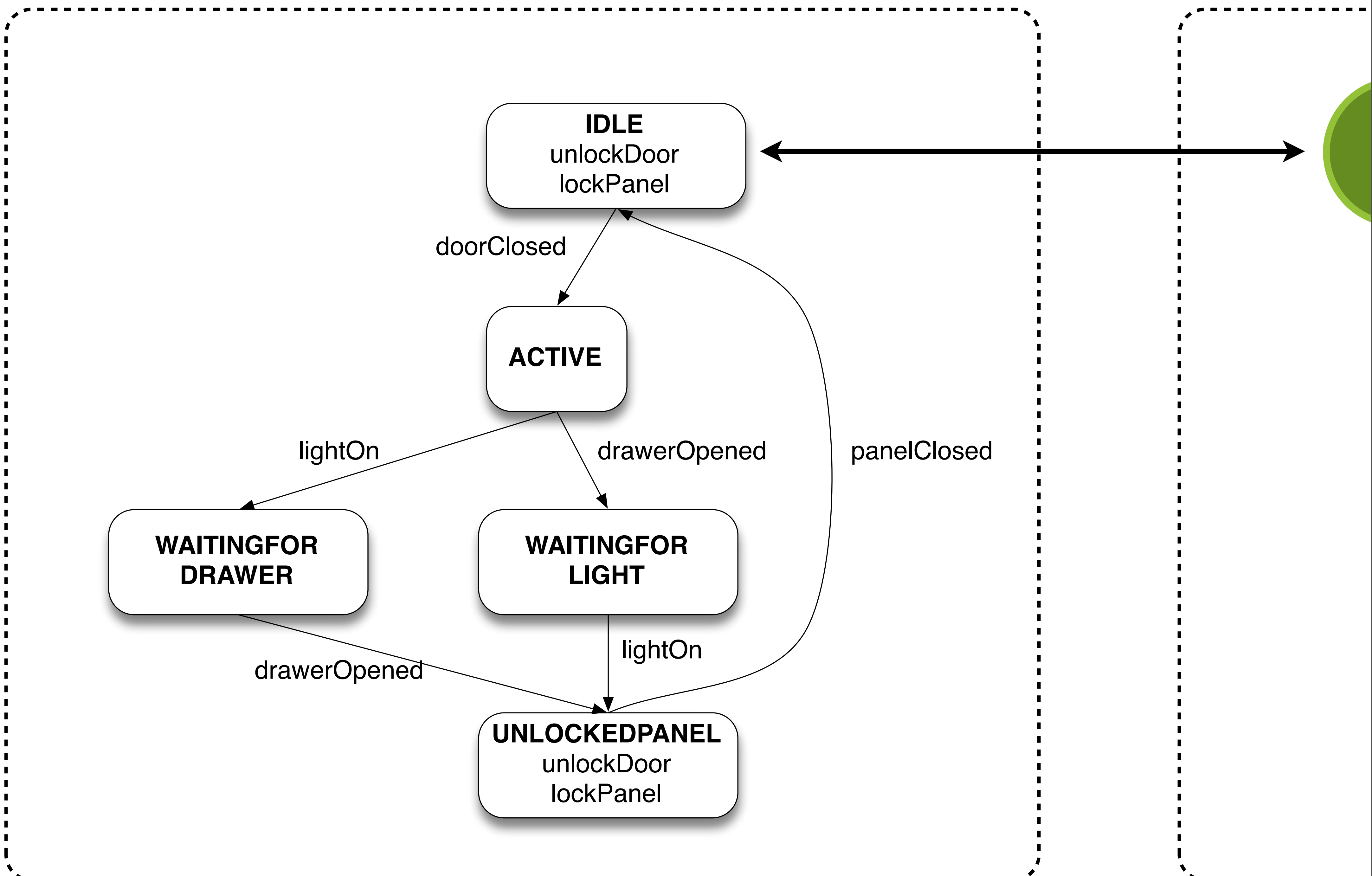
UNIFY CONCEPTS



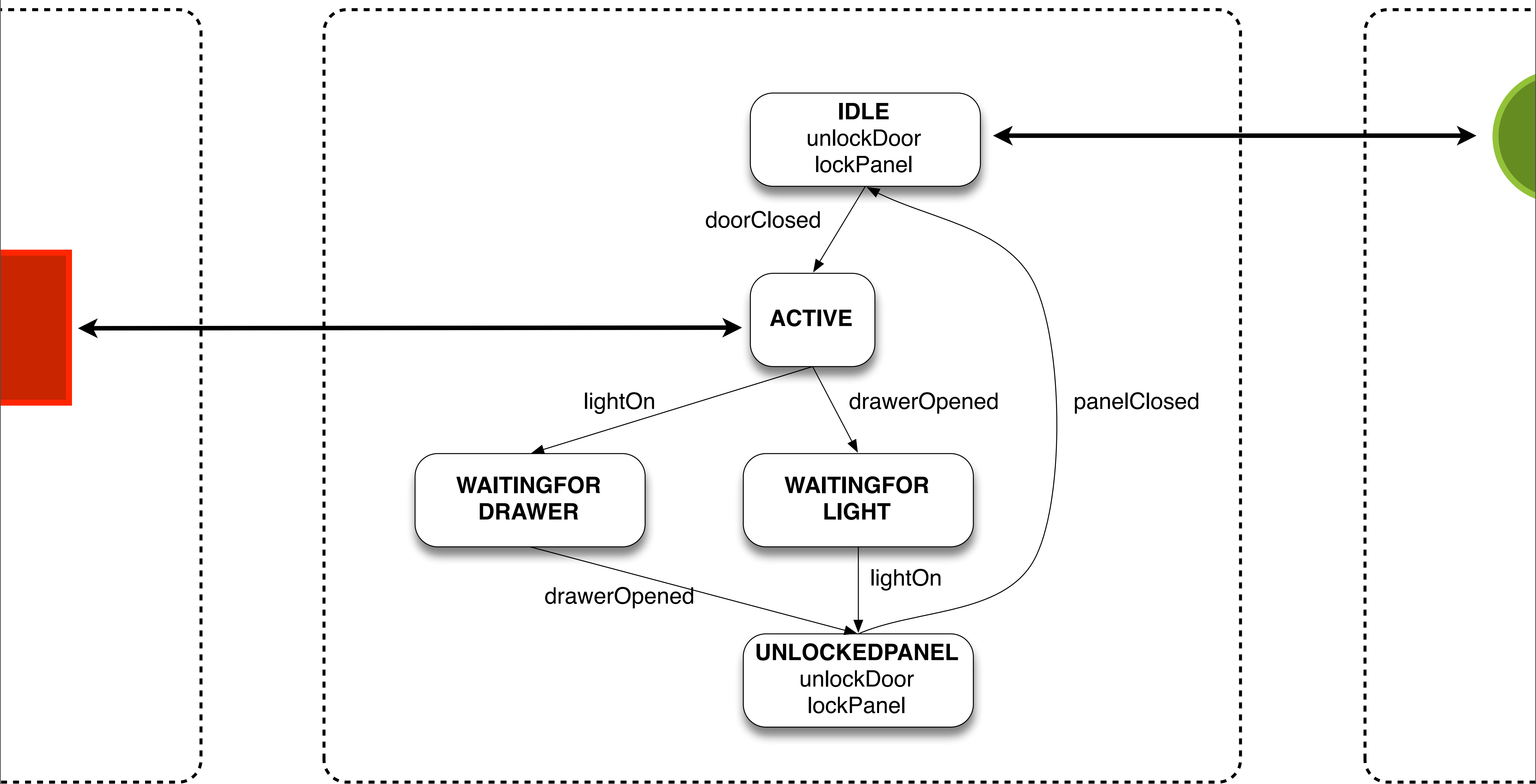
UNIFY CONCEPTS



UNIFY CONCEPTS



UNIFY CONCEPTS



CONCLUSIONS

CONCLUSIONS

- Language of Languages: An experimental language workbench that unifies concepts expressed across different notations

CONCLUSIONS

- Language of Languages: An experimental language workbench that unifies concepts expressed across different notations
- Implemented using ideas of Language Elements, Concepts and Language Definitions

CONCLUSIONS

- Language of Languages: An experimental language workbench that unifies concepts expressed across different notations
- Implemented using ideas of Language Elements, Concepts and Language Definitions
- Available now from www.languageoflanguages.org

CONCLUSIONS

- Language of Languages: An experimental language workbench that unifies concepts expressed across different notations
- Implemented using ideas of Language Elements, Concepts and Language Definitions
- Available now from www.languageoflanguages.org
- Contact Jamie Douglass (jamie.douglass@boeing.com) or Nicholas Chen (nchen@illinois.edu)