

Abstract Syntax and Name Resolution

Tijs van der Storm



Centrum Wiskunde & Informatica

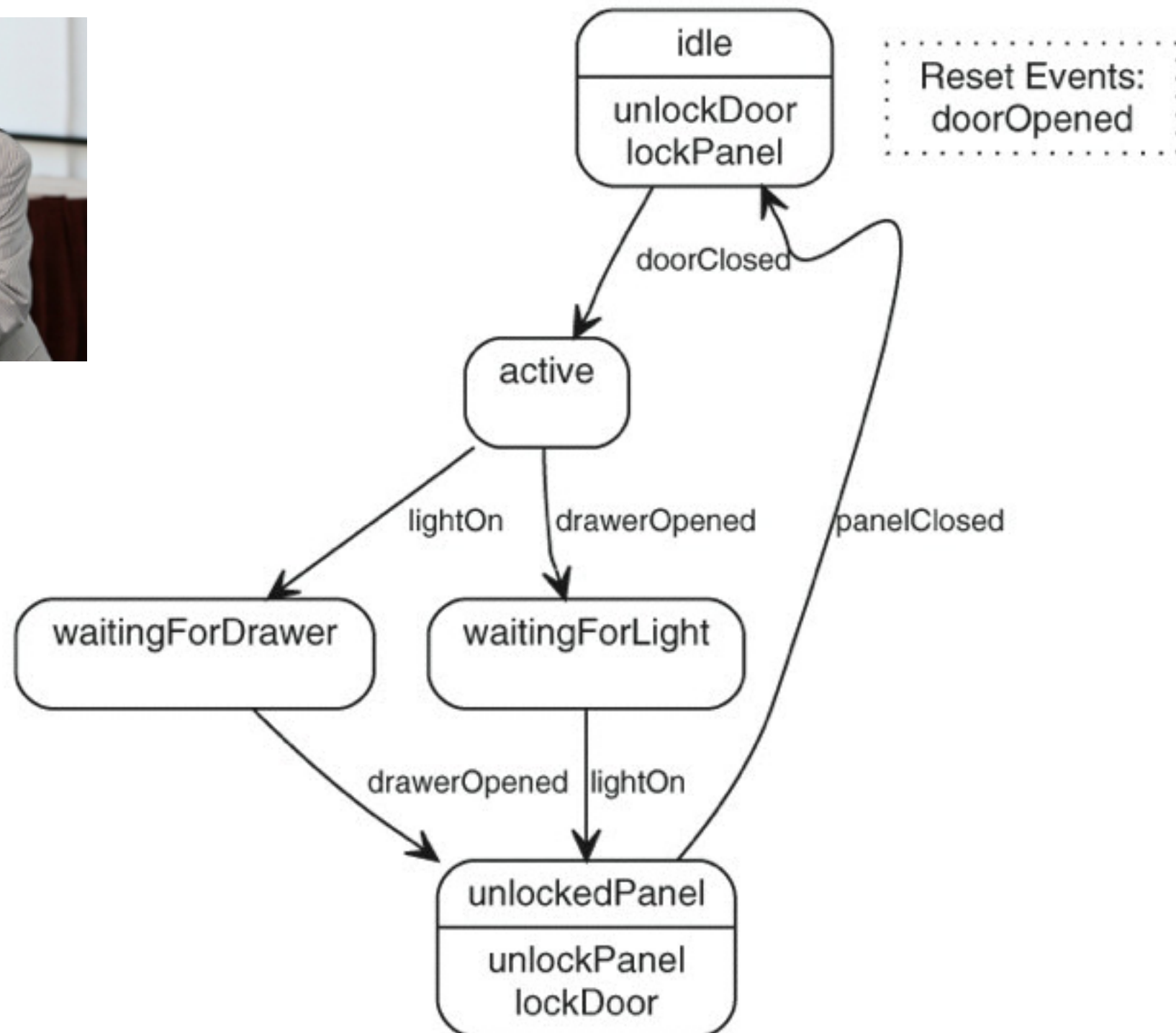


university of
groningen

Recap

- Context-free grammars to define syntax
- Parser generator: Grammar \rightarrow Parser
- Parser: Text \rightarrow Concrete syntax tree
- Today:
 - abstract syntax representations
 - name resolution

Example: state machines



events

```
doorClosed D1CL  
drawerOpened D2OP  
lightOn L1ON  
doorOpened D1OP  
panelClosed PNCL
```

end

resetEvents

```
doorOpened
```

end

commands

```
unlockPanel PNUL  
lockPanel PNLK  
lockDoor D1LK  
unlockDoor D1UL
```

end

state idle

```
actions {unlockDoor lockPanel}  
doorClosed => active
```

end

state active

```
drawerOpened => waitingForLight  
lightOn => waitingForDrawer
```

end

state waitingForLight

```
lightOn => unlockedPanel
```

end

state waitingForDrawer

```
drawerOpened => unlockedPanel
```

end

state unlockedPanel

```
actions {unlockPanel lockDoor}  
panelClosed => idle
```

end

Grammars

```
module Syntax  
  extend Lang::std::Layout;
```

standard
Layout

start
symbol

```
  start syntax Controller =  
    controller:
```

production
label

```
      Events events  
      ResetEvents? resets  
      Commands? commands  
      State+ states;
```

subelement
labels

enable
folding

```
  syntax Events  
    = @Foldable "events" Event* "end";  
  syntax ResetEvents  
    = @Foldable "resetEvents" Id* "end";  
  syntax Commands  
    = @Foldable "commands" Command* "end";
```

Lexical syntax

lexicals don't
get layout

follow
restriction

character
class

```
lexical Id  
= ([a-zA-Z][a-zA-Z0-9_]* !>> [a-zA-Z0-9_])  
\ Reserved ;
```

keyword
reservation

```
keyword Reserved  
= "events"  
| "end"  
| "resetEvents"  
| "state"  
| "actions" ;
```

keyword
class

[illegible]

```
low(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)])))]),[char(68),appl(regular(\iter-star(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)])))],[char(50),char(79),char(80)]][@loc=|project://MissGrant/input/missgrant.ctll
(39,3,<3,15>,<3,18>)]])[@loc=|project://MissGrant/input/missgrant.ctll(38,4,<3,14>,<3,18>)]])[@loc=|project://MissGrant/input/missgrant.ctll
(38,4,<3,14>,<3,18>)]])[@loc=|project://MissGrant/input/missgrant.ctll(25,17,<3,1>,<3,18>)],appl(prod(layouts("Standard"),[conditional(\iter-
star(sort("WhitespaceOrComment")),{\not-follow(\char-class([range(9,10),range(12,13),range(32,32)])),\not-follow(lit("/")})}],{}),
[appl(regular(\iter-star(sort("WhitespaceOrComment")))],[appl(prod(label("whitespace",sort("WhitespaceOrComment")),[lex("Whitespace")],{}),
[appl(prod(lex("Whitespace"),[\char-class([range(9,10),range(12,13),range(32,32)]),{}], [char(10)]][@loc=|project://MissGrant/input/
missgrant.ctll(42,1,<3,18>,<4,0>)]])[@loc=|project://MissGrant/input/missgrant.ctll
(42,1,<3,18>,<4,0>)],appl(prod(label("whitespace",sort("WhitespaceOrComment")), [lex("Whitespace")],{}), [appl(prod(lex("Whitespace"), [\char-
class([range(9,10),range(12,13),range(32,32)]),{}], [char(32)]][@loc=|project://MissGrant/input/missgrant.ctll(43,1,<4,0>,<4,1>)]])[@loc=|
project://MissGrant/input/missgrant.ctll(43,1,<4,0>,<4,1>)]])[@loc=|project://MissGrant/input/missgrant.ctll(42,2,<3,18>,<4,1>)]])[@loc=|
project://MissGrant/input/missgrant.ctll(42,2,<3,18>,<4,1>)],appl(prod(label("event",sort("Event")),
[label("name",lex("Id")),layouts("Standard"),label("token",lex("Id"))],{}), [appl(prod(lex("Id"), [conditional(seq([\char-
class([range(65,90),range(97,122)]),conditional(\iter-star(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)])),{\not-
follow(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)]))})]), {delete(keywords("Reserved"))})}],{}),
[appl(regular(seq([\char-class([range(65,90),range(97,122)]),conditional(\iter-star(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)])),{\not-follow(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)]))})]), [char(108),appl(regular(\iter-star(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)]))), [char(105),char(103),char(104),char(116),char(79),char(110)]][@loc=|
project://MissGrant/input/missgrant.ctll(45,6,<4,2>,<4,8>)]])[@loc=|project://MissGrant/input/missgrant.ctll(44,7,<4,1>,<4,8>)]])[@loc=|
project://MissGrant/input/missgrant.ctll(44,7,<4,1>,<4,8>)],appl(prod(layouts("Standard"),[conditional(\iter-
star(sort("WhitespaceOrComment")),{\not-follow(\char-class([range(9,10),range(12,13),range(32,32)])),\not-follow(lit("/")})}],{}),
[appl(regular(\iter-star(sort("WhitespaceOrComment")))], [appl(prod(label("whitespace",sort("WhitespaceOrComment")), [lex("Whitespace")],{}),
[appl(prod(lex("Whitespace"), [\char-class([range(9,10),range(12,13),range(32,32)]),{}], [char(32)]][@loc=|project://MissGrant/input/
missgrant.ctll(51,1,<4,8>,<4,9>)]])[@loc=|project://MissGrant/input/missgrant.ctll(51,1,<4,8>,<4,9>)]])[@loc=|project://MissGrant/input/
missgrant.ctll(51,1,<4,8>,<4,9>)]])[@loc=|project://MissGrant/input/missgrant.ctll(51,1,<4,8>,<4,9>)],appl(prod(lex("Id"),
[conditional(seq([\char-class([range(65,90),range(97,122)]),conditional(\iter-star(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)])),{\not-follow(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)]))})]), {delete(keywords("Reserved"))})],{}), [appl(regular(seq([\char-
class([range(65,90),range(97,122)]),conditional(\iter-star(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)])),{\not-
follow(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)]))})]), [char(76),appl(regular(\iter-star(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)]))), [char(49),char(79),char(78)]][@loc=|project://MissGrant/input/missgrant.ctll
(53,3,<4,10>,<4,13>)]])[@loc=|project://MissGrant/input/missgrant.ctll(52,4,<4,9>,<4,13>)]])[@loc=|project://MissGrant/input/missgrant.ctll
(52,4,<4,9>,<4,13>)]])[@loc=|project://MissGrant/input/missgrant.ctll(44,12,<4,1>,<4,13>)],appl(prod(layouts("Standard"),[conditional(\iter-
star(sort("WhitespaceOrComment")),{\not-follow(\char-class([range(9,10),range(12,13),range(32,32)])),\not-follow(lit("/")})}],{}),
[appl(regular(\iter-star(sort("WhitespaceOrComment")))], [appl(prod(label("whitespace",sort("WhitespaceOrComment")), [lex("Whitespace")],{}),
[appl(prod(lex("Whitespace"), [\char-class([range(9,10),range(12,13),range(32,32)]),{}], [char(10)]][@loc=|project://MissGrant/input/
missgrant.ctll(56,1,<4,13>,<5,0>)]])[@loc=|project://MissGrant/input/missgrant.ctll
(56,1,<4,13>,<5,0>)],appl(prod(label("whitespace",sort("WhitespaceOrComment")), [lex("Whitespace")],{}), [appl(prod(lex("Whitespace"), [\char-
class([range(9,10),range(12,13),range(32,32)]),{}], [char(32)]][@loc=|project://MissGrant/input/missgrant.ctll(57,1,<5,0>,<5,1>)]])[@loc=|
project://MissGrant/input/missgrant.ctll(57,1,<5,0>,<5,1>)]])[@loc=|project://MissGrant/input/missgrant.ctll(56,2,<4,13>,<5,1>)]])[@loc=|
project://MissGrant/input/missgrant.ctll(56,2,<4,13>,<5,1>)],appl(prod(label("event",sort("Event")),
[label("name",lex("Id")),layouts("Standard"),label("token",lex("Id"))],{}), [appl(prod(lex("Id"), [conditional(seq([\char-
class([range(65,90),range(97,122)]),conditional(\iter-star(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)])),{\not-
follow(\char-class([range(48,57),range(65,90),range(95,95),range(97,122)]))})]), {delete(keywords("Reserved"))})}],{}),
[appl(regular(seq([\char-class([range(65,90),range(97,122)]),conditional(\iter-star(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)])),{\not-follow(\char-
class([range(48,57),range(65,90),range(95,95),range(97,122)]))})]), [char(100),appl(regular(\iter-star(\char-
```


non-terminals
map to data type

abstract syntax

```
data Controller
  = controller(list[Event] events,
               list[str] resets,
               list[Command] commands,
               list[State] states);
```

regulars (+/*/?)
map to lists

productions to
constructors

```
data State
  = state(str name,
          list[str] actions,
          list[Transition] transitions);
```

lexicals to str/
int/real/bool

```
data Command    = command(str name, str token);
data Event      = event(str name, str token);
data Transition  = transition(str event, str state);
```

“implode”

controller{

```
event("doorClosed", "D1CL"),
event("drawerOpened", "D2OP"),
event("lightOn", "L1ON"),
event("doorOpened", "D1OP"),
event("panelClosed", "PNCL")
```

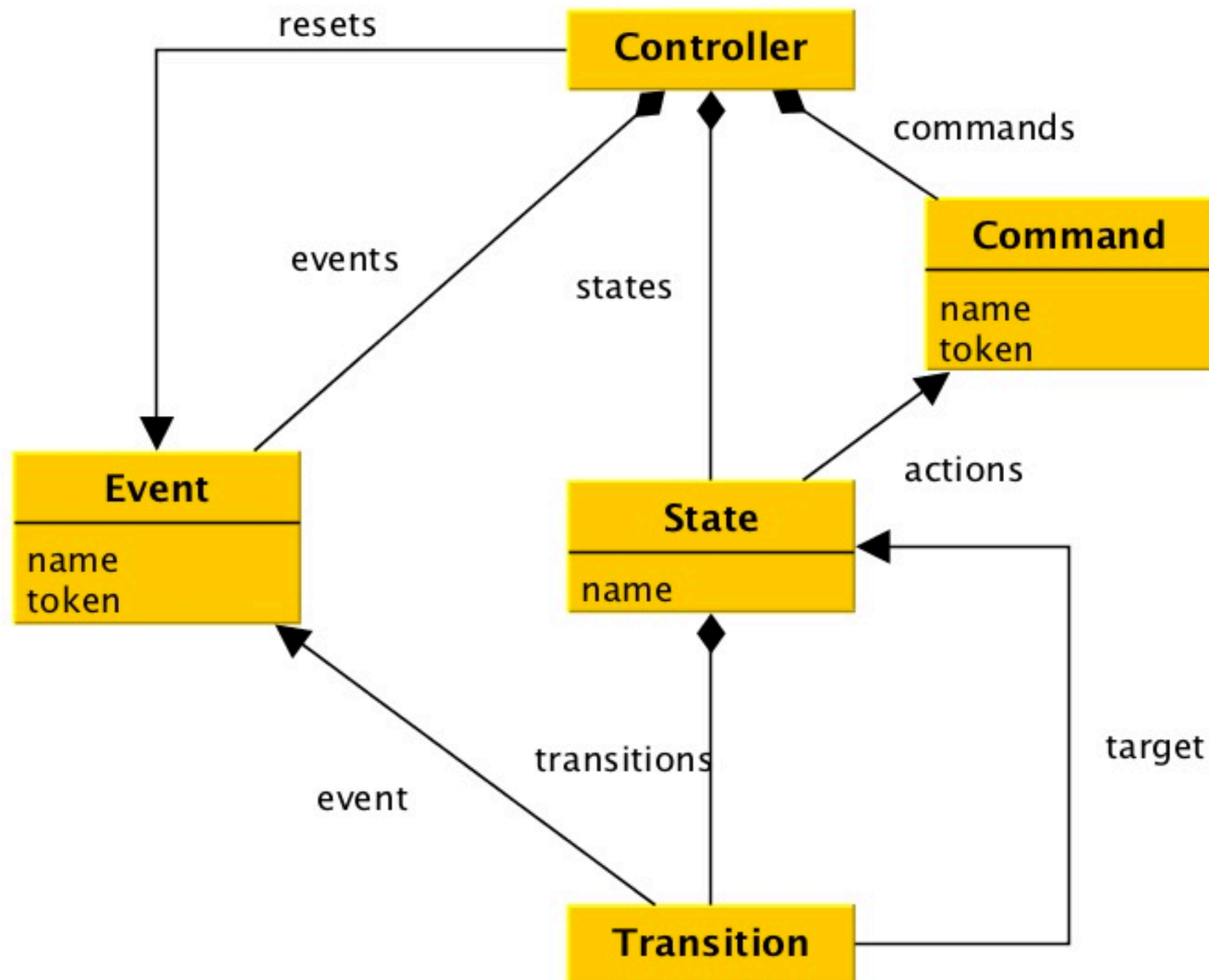
```
},
["doorOpened"],
[
command("unlockPanel", "PNUL"),
command("lockPanel", "PNLK"),
command("lockDoor", "D1LK"),
command("unlockDoor", "D1UL")
```

```
],
[
state(
"idle",
["unlockDoor", "lockPanel"],
[transition("doorClosed", "active")],
state(
"active",
[],
[
transition("drawerOpened", "waitingForLight"),
transition("lightOn", "waitingForDrawer")
]),
state(
"waitingForLight",
[],
[transition("lightOn", "unlockedPanel")],
state(
"waitingForDrawer",
[],
[transition("drawerOpened", "unlockedPanel")],
state(
"unlockedPanel",
["unlockPanel", "lockDoor"],
[transition("panelClosed", "idle")]
)
)
)
}
```

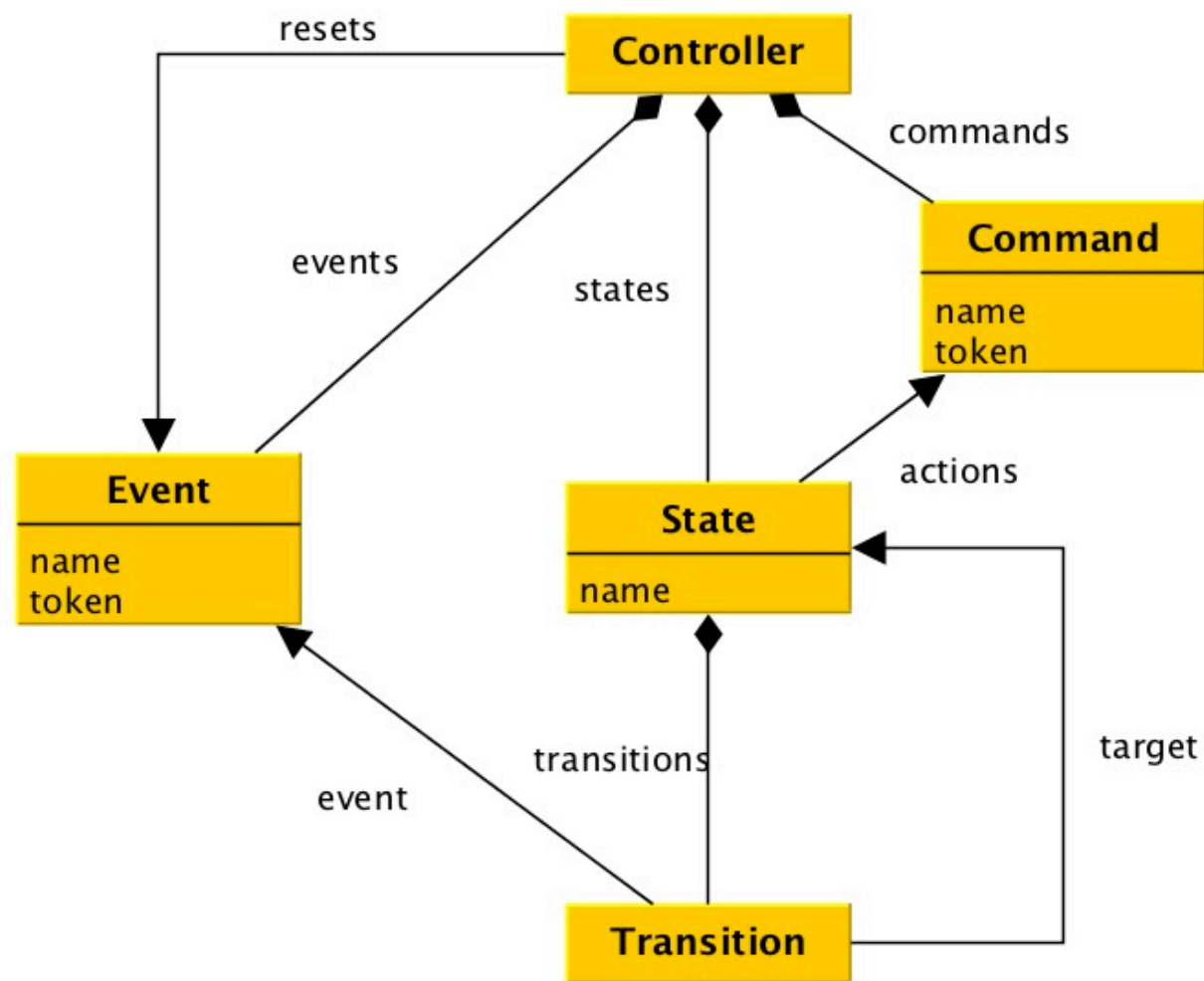
Meta Models

- Algebraic data types (“data Expr = ...”)
 - Define type of abstract syntax *trees*
- Meta Models are UML-style class diagrams
 - Define type of object *models*
- Both have benefits and disadvantages
- => Technological space (grammarware, modelware)

State machine meta model



References: explicit or using symbolic names



```
data Controller
    = controller(list[Event] events,
                 list[str] resets,
                 list[Command] commands,
                 list[State] states);
```

```
data State
    = state(str name,
            list[str] actions,
            list[Transition] transitions);
```

```
data Command
    = command(str name, str token);
```

```
data Event
    = event(str name, str token);
```

```
data Transition
    = transition(str event, str state);
```


events

doorClosed D1CL
drawerOpened D2OP
lightOn L1ON
doorOpened D1OP
panelClosed PNCL

end

resetEvents

doorOpened

end

commands

unlockPanel PNUL
lockPanel PNLK
lockDoor D1LK
unlockDoor D1UL

end

state idle

actions {unlockDoor lockPanel}
doorClosed => active

end

state active

drawerOpened => waitingForLight
lightOn => waitingForDrawer

end

state waitingForLight

lightOn => unlockedPanel

end

state waitingForDrawer

drawerOpened => unlockedPanel

end

state unlockedPanel

actions {unlockPanel lockDoor}
panelClosed => idle

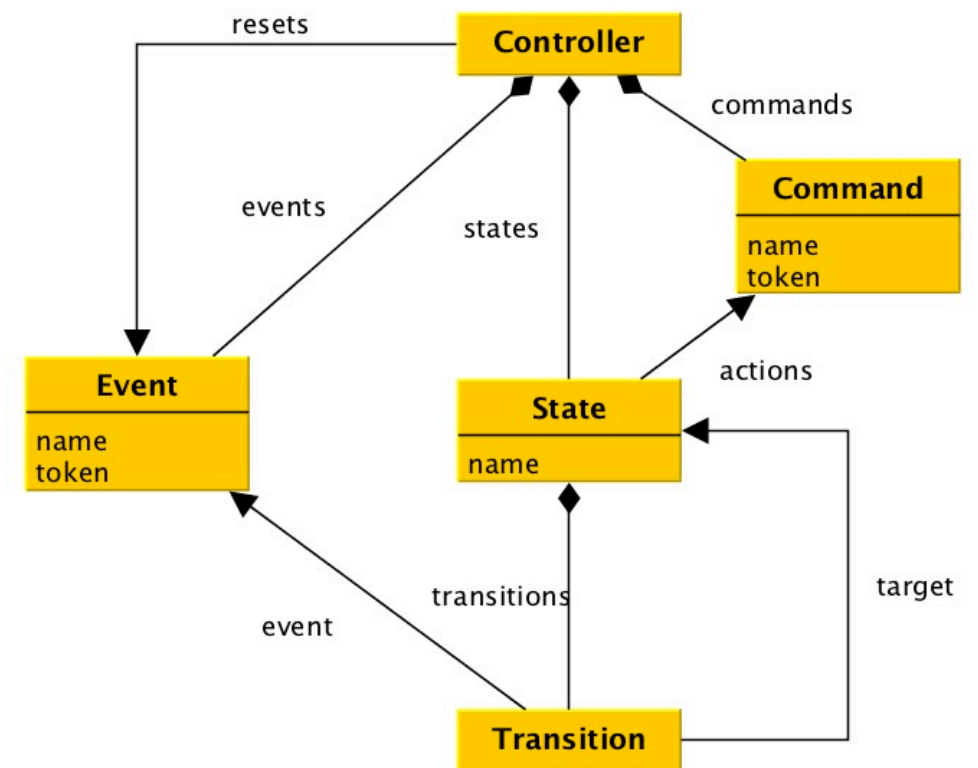
end

Names

- **defining occurrence**: declaration, or entity, introduction of a name
 - Example: states, events, commands
- **use occurrence**: reference, variable use, etc.
 - Example: reference to event/target state in transition, reference to commands in states

Name resolution

- Finding which use occurrences refer to which defining occurrences
- Recovering the referential structure represented in meta models (roughly)
- IOW: defining the **scoping** rules of a language



Name resolution in Rascal

- Name occurrences (both defining and using) can be identified by **source locations**
- Source locations are unique: no two name occurrences can overlap.
- So we can use source locations as **identities** to represent the referential structure of a program.

Source locations

Scheme

Authority

Path

Offset

Length

|project://mutyphonql/src/example.mutql|(45,8,<6,2>,<6,10>)

Start line
and column

End line and
column

NB: clicking on a source
location jumps to exact
location in an editor

Reference graphs

- Using source locations as identities of concepts in a language, the referential structure is typically represented as **reference graph**
- In Rascal: `rel[loc, loc]`
- This can be used to:
 - find naming errors and warnings
 - provide jump-to-definition IDE support
 - use in a compiler to find the declaration of an identifier
 - visualization, etc.

Next up

- Live coding a simpler state machine example
 - abstract syntax, implode, name analysis
- Providing an introduction to the lab assignment skeleton code for this week.