I want my live programming environment!

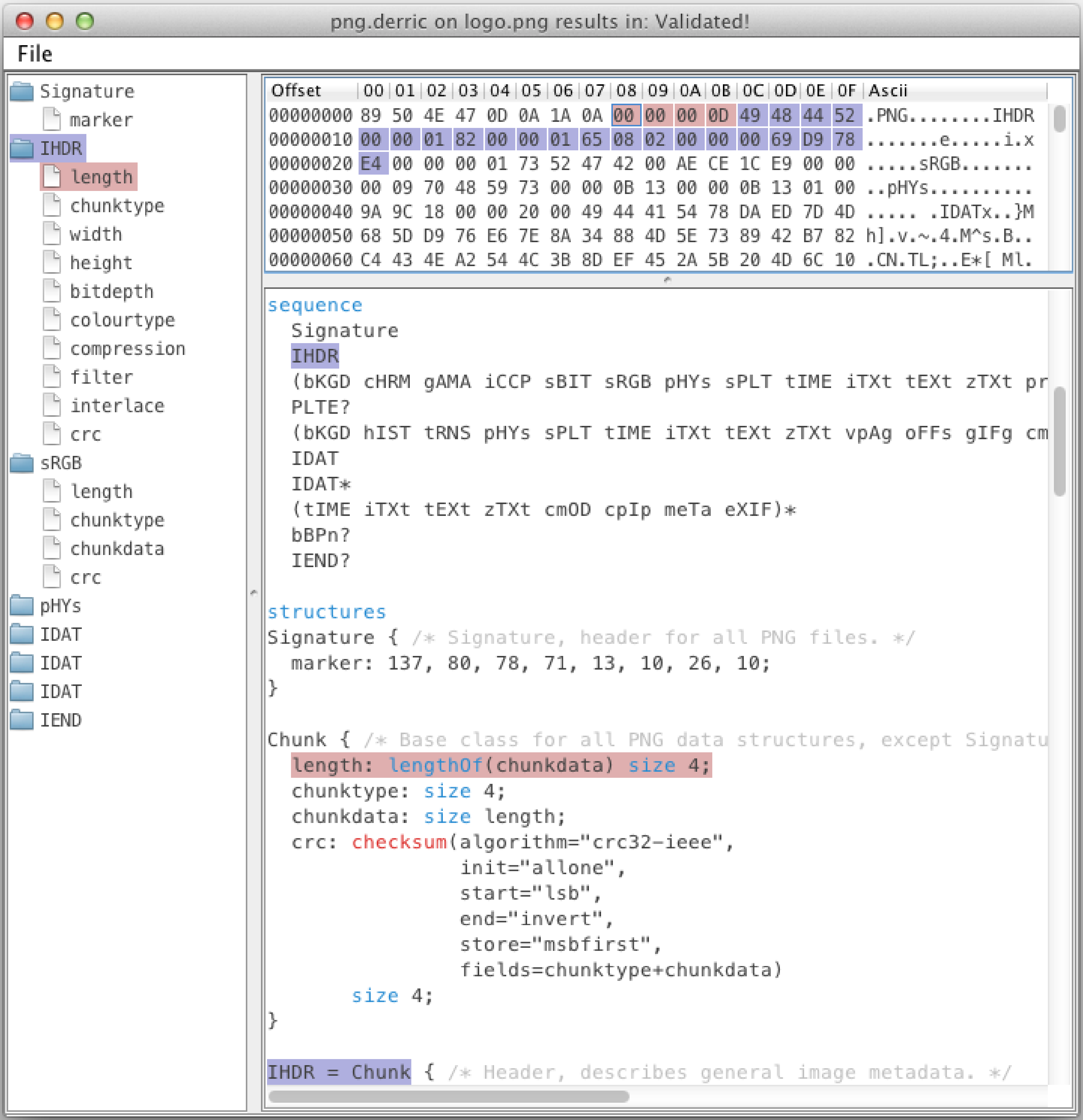
Tijs van der Storm

CWI Centrum Wiskunde & Informatica

University of Amsterdam

Context

- External DSLs
- Language workbench: Rascal
- Want: live DSL environments



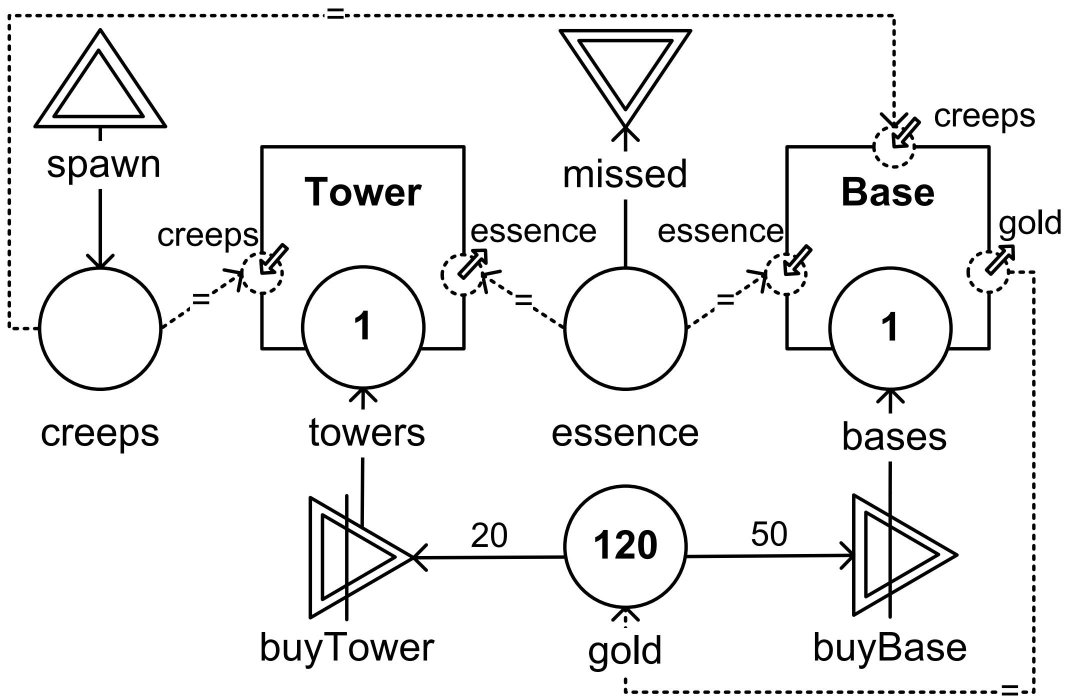
```
Signature
  marker
IHDR
  length
  chunktype
  width
  height
  bitdepth
  colourtype
  compression
  filter
  interlace
  crc
sRGB
  length
  chunktype
  chunkdata
  crc
pHYs
IDAT
IDAT
IDAT
IEND
```

```
Offset   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Ascii
00000000 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 .PNG........IHDR
00000010 00 00 01 82 00 00 01 65 08 02 00 00 00 69 D9 78 .......e.....i.x
00000020 E4 00 00 00 01 73 52 47 42 00 AE CE 1C E9 00 00 .....sRGB.......
00000030 00 09 70 48 59 73 00 00 0B 13 00 00 0B 13 01 00 ..pHYs..........
00000040 9A 9C 18 00 00 20 00 49 44 41 54 78 DA ED 7D 4D ..... .IDATx..}M
00000050 68 5D D9 76 E6 7E 8A 34 88 4D 5E 73 89 42 B7 82 h].v.~.4.M^s.B..
00000060 C4 43 4E A2 54 4C 3B 8D EF 45 2A 5B 20 4D 6C 10 .CN.TL;..E*[ Ml.
```

```
sequence
  Signature
  IHDR
  (bKGD cHRM gAMA iCCP sBIT sRGB pHYs sPLT tIME iTXt tEXt zTXt pr
  PLTE?
  (bKGD hIST tRNS pHYs sPLT tIME iTXt tEXt zTXt vpAg oFFs gIFg cm
  IDAT
  IDAT*
  (tIME iTXt tEXt zTXt cmOD cpIp meTa eXIF)*
  bBPn?
  IEND?

structures
Signature { /* Signature, header for all PNG files. */
  marker: 137, 80, 78, 71, 13, 10, 26, 10;
}

Chunk { /* Base class for all PNG data structures, except Signatu
  length: lengthOf(chunkdata) size 4;
  chunktype: size 4;
  chunkdata: size length;
  crc: checksum(algorithm="crc32-ieee",
                init="allone",
                start="lsb",
                end="invert",
                store="msbfirst",
                fields=chunktype+chunkdata)
       size 4;
}

IHDR = Chunk { /* Header, describes general image metadata. */
```

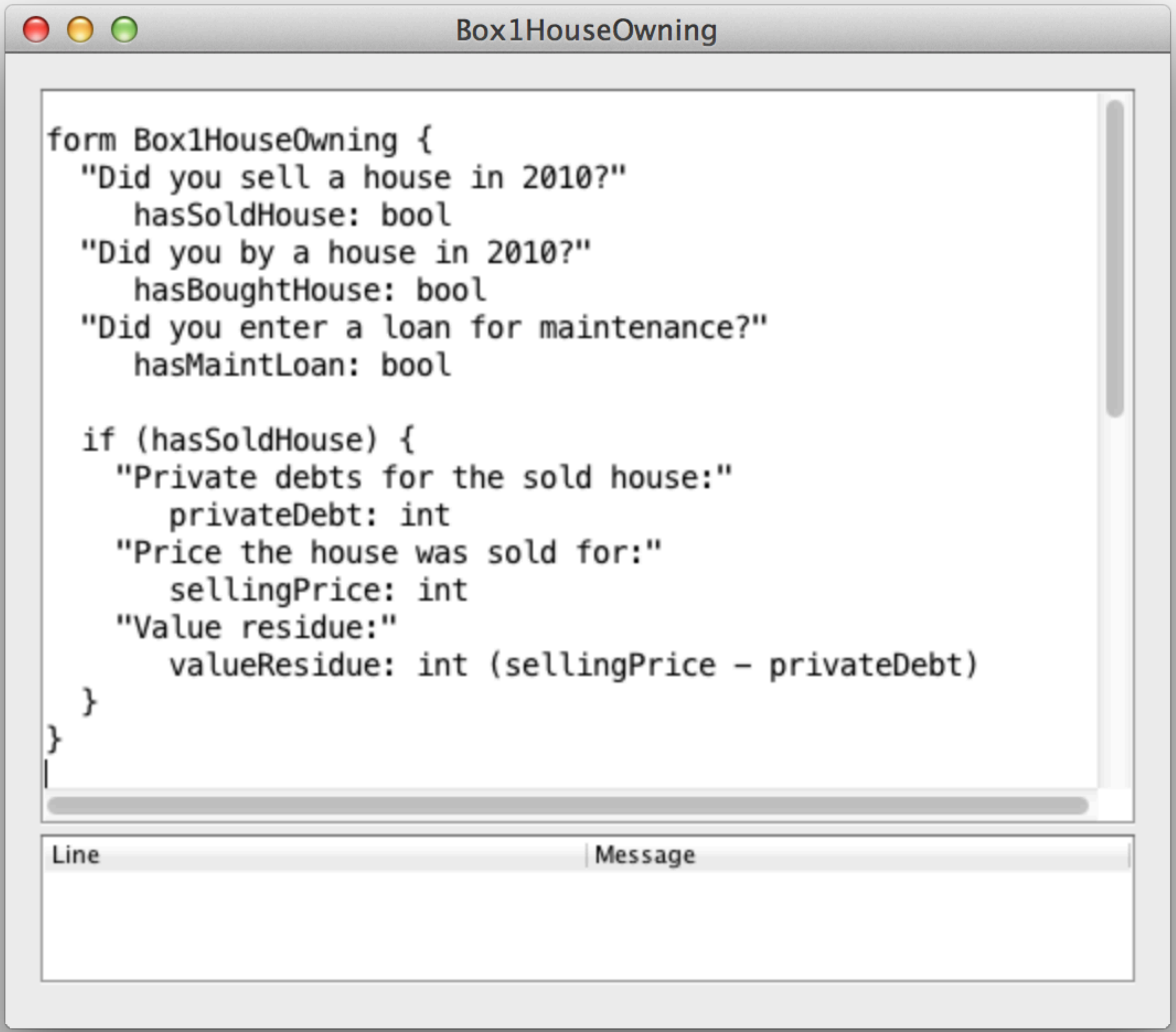
Trinity, an IDE for the Matrix

Micro Machinations

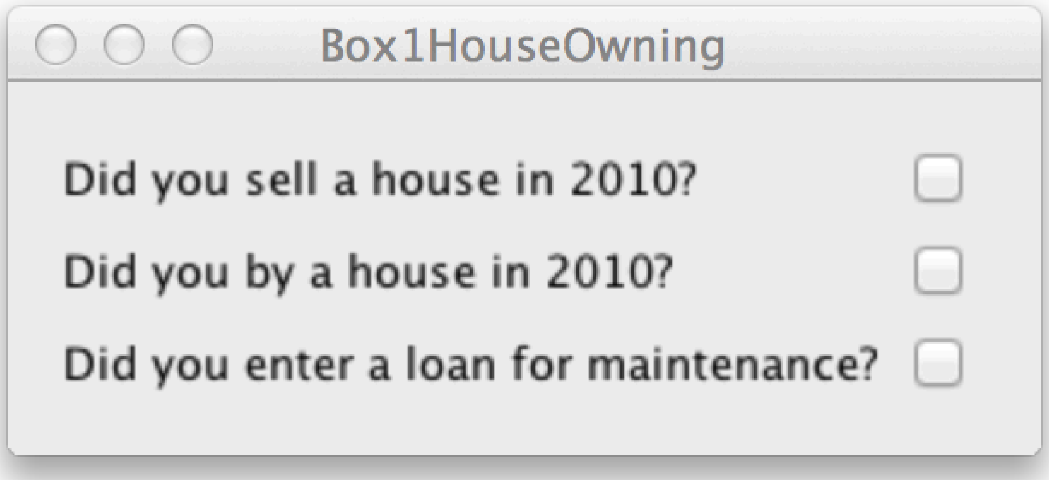


Van Rozen, Dormans, Adapting game mechanics with Micro-Machinations, FDG'14

Live QL



```
form Box1HouseOwning {
  "Did you sell a house in 2010?"
     hasSoldHouse: bool
  "Did you by a house in 2010?"
     hasBoughtHouse: bool
  "Did you enter a loan for maintenance?"
     hasMaintLoan: bool

  if (hasSoldHouse) {
    "Private debts for the sold house:"
       privateDebt: int
    "Price the house was sold for:"
       sellingPrice: int
    "Value residue:"
       valueResidue: int (sellingPrice - privateDebt)
  }
}
```

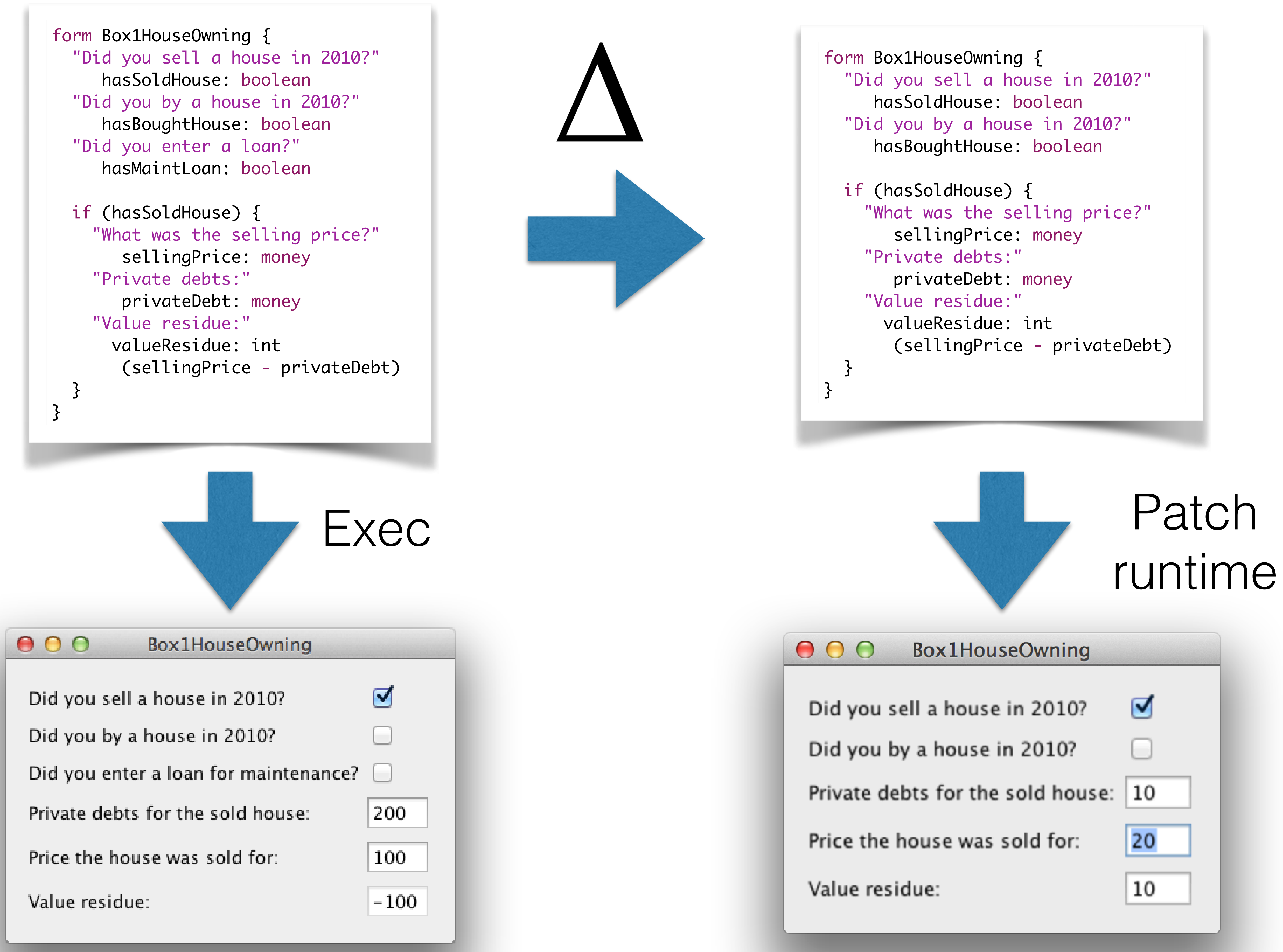


Did you sell a house in 2010? ☐

Did you by a house in 2010? ☐

Did you enter a loan for maintenance? ☐

van der Storm, *Semantic Deltas for Live DSL Environments*, LIVE'13



```
form Box1HouseOwning {
  "Did you sell a house in 2010?"
     hasSoldHouse: boolean
  "Did you by a house in 2010?"
     hasBoughtHouse: boolean
  "Did you enter a loan?"
     hasMaintLoan: boolean

  if (hasSoldHouse) {
    "What was the selling price?"
       sellingPrice: money
    "Private debts:"
       privateDebt: money
    "Value residue:"
      valueResidue: int
       (sellingPrice - privateDebt)
  }
}
```

```
form Box1HouseOwning {
  "Did you sell a house in 2010?"
     hasSoldHouse: boolean
  "Did you by a house in 2010?"
     hasBoughtHouse: boolean

  if (hasSoldHouse) {
    "What was the selling price?"
       sellingPrice: money
    "Private debts:"
       privateDebt: money
    "Value residue:"
      valueResidue: int
       (sellingPrice - privateDebt)
  }
}
```

Box1HouseOwning

Did you sell a house in 2010?	✓
Did you by a house in 2010?	
Did you enter a loan for maintenance?	
Private debts for the sold house:	200
Price the house was sold for:	100
Value residue:	-100

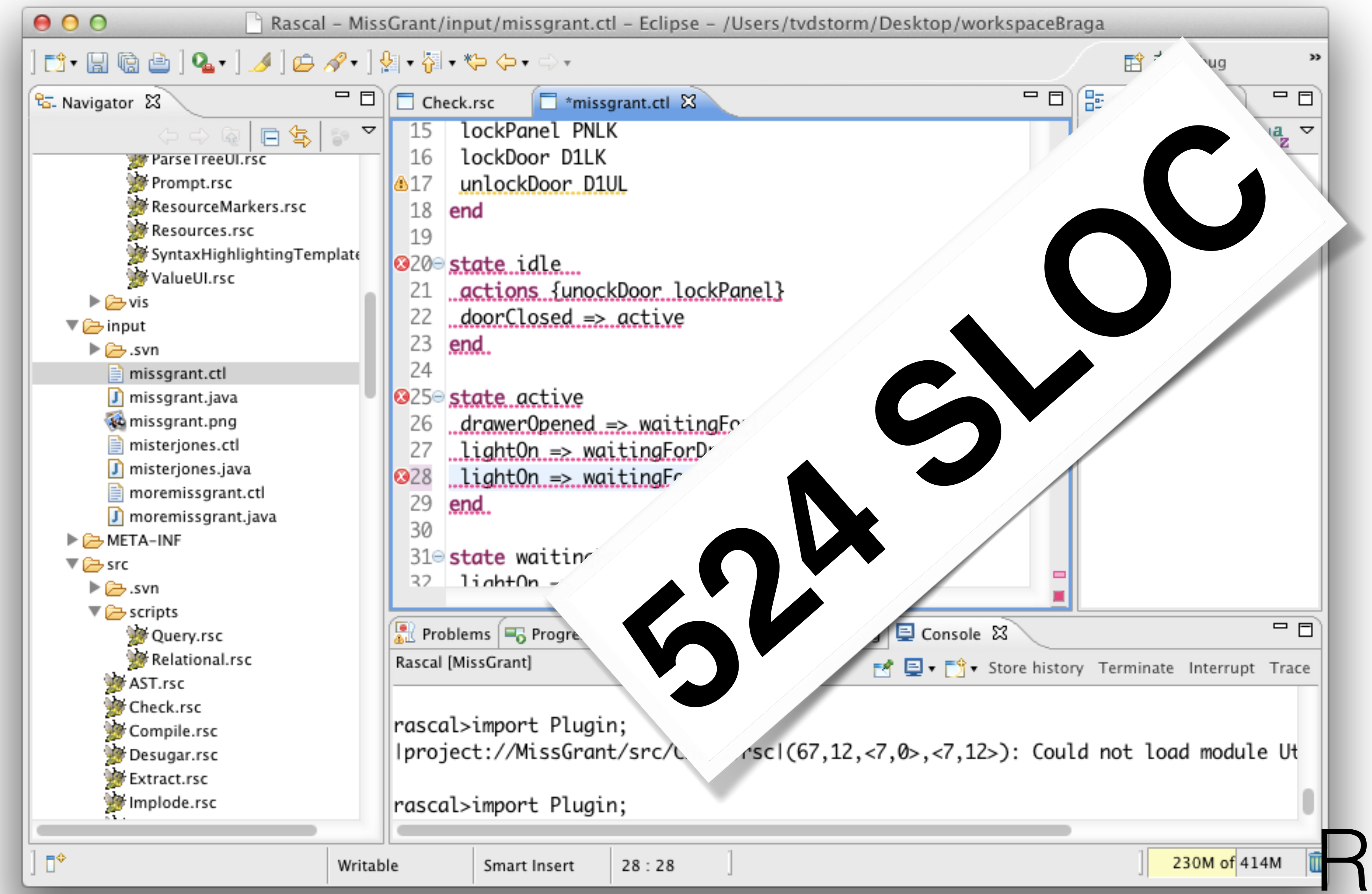
Box1HouseOwning

Did you sell a house in 2010?	✓
Did you by a house in 2010?	
Private debts for the sold house:	10
Price the house was sold for:	20
Value residue:	10

Language workbenches

- Encoding language designs of the past…
- In generic, reusable and limiting tools ;)
- Language engineering vs. “PL”
- Our approach: Rascal
- FP for meta programming

State machine DSL in Rascal



```
lockPanel PNLK
lockDoor D1LK
unlockDoor D1UL
end

state idle
actions {unockDoor lockPanel}
doorClosed => active
end

state active
drawerOpened => waitingFo
lightOn => waitingForD
lightOn => waitingFo
end

state waitin
lightOn -
```

```
rascal>import Plugin;
|project://MissGrant/src/(        rsc|(67,12,<7,0>,<7,12>): Could not load module Ut

rascal>import Plugin;
```

524 SLOC

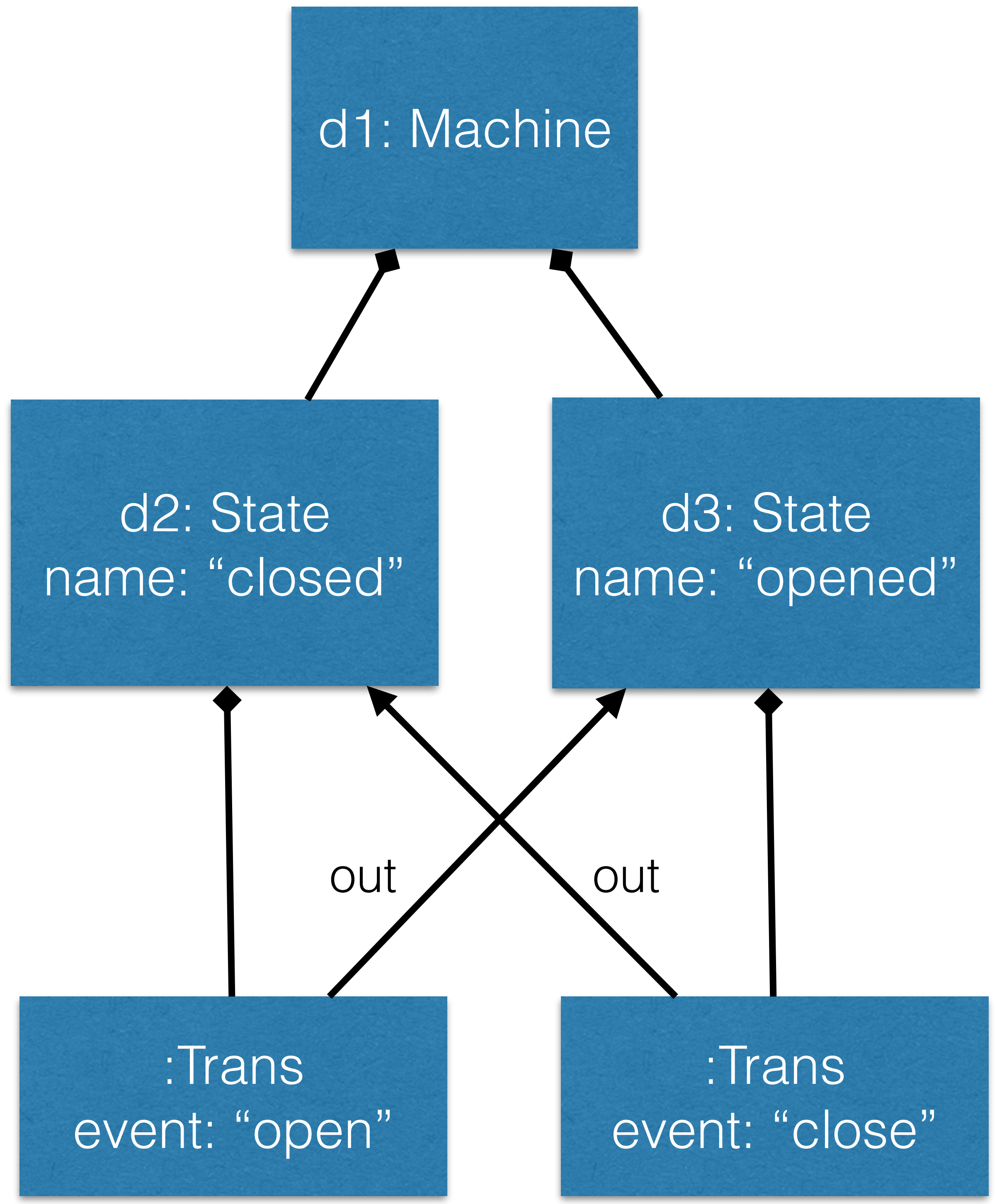
Concrete syntax
Abstract syntax
Unparse
Desugaring
Checking
Outline
Hyperlinking
Compilation
Visual simulation
Rename refactoring
Parallel merge

Goal: generic tools to bridge the gulf of evaluation

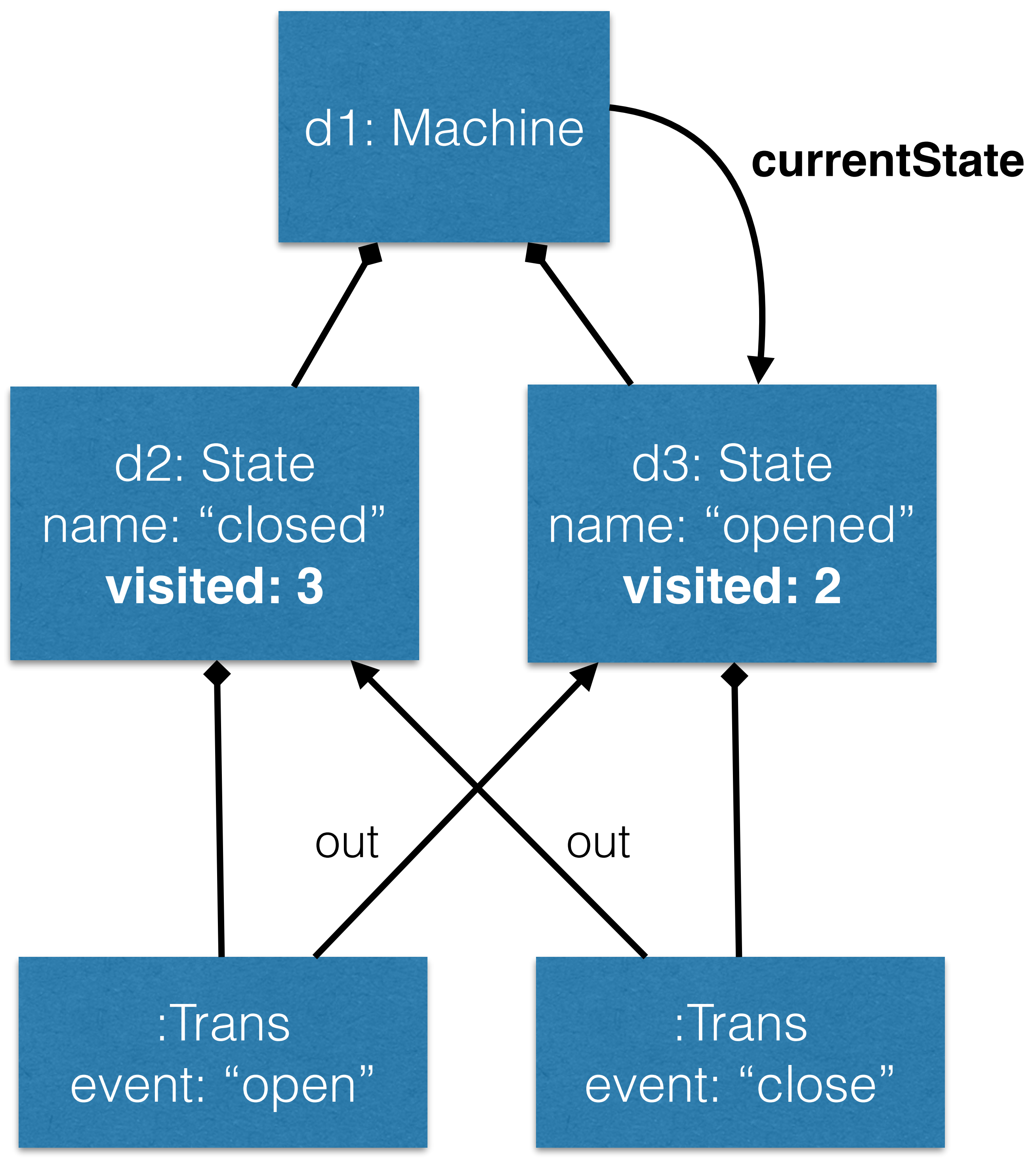
Framework

- “Programs as models”
- Assume relation between static model and runtime model
- Generic *diff* to obtain semantic deltas
- Generically patch the runtime model
- Specialize patch where needed to migrate runtime state

```
machine doors d1
  state closed d2
    open => opened u1

  state opened d3
    close => closed u2

end
```



```
machine doors d1
  state closed d2
    open => opened u1

  state opened d3
    close => closed u2


end
```



```
machine doors d1
  state closed d2
    open => opened u1

  state opened d3
    close => closed u2

end
```

```
machine doors d4
  state closed d5
    open => opened u3
    lock => locked u4

  state opened d6
    close => closed u5

  state locked d7
    unlock => closed u6

end
```

diff (

```
machine doors d1
  state closed d2
    open => opened u1

  state opened d3
    close => closed u2

end
```

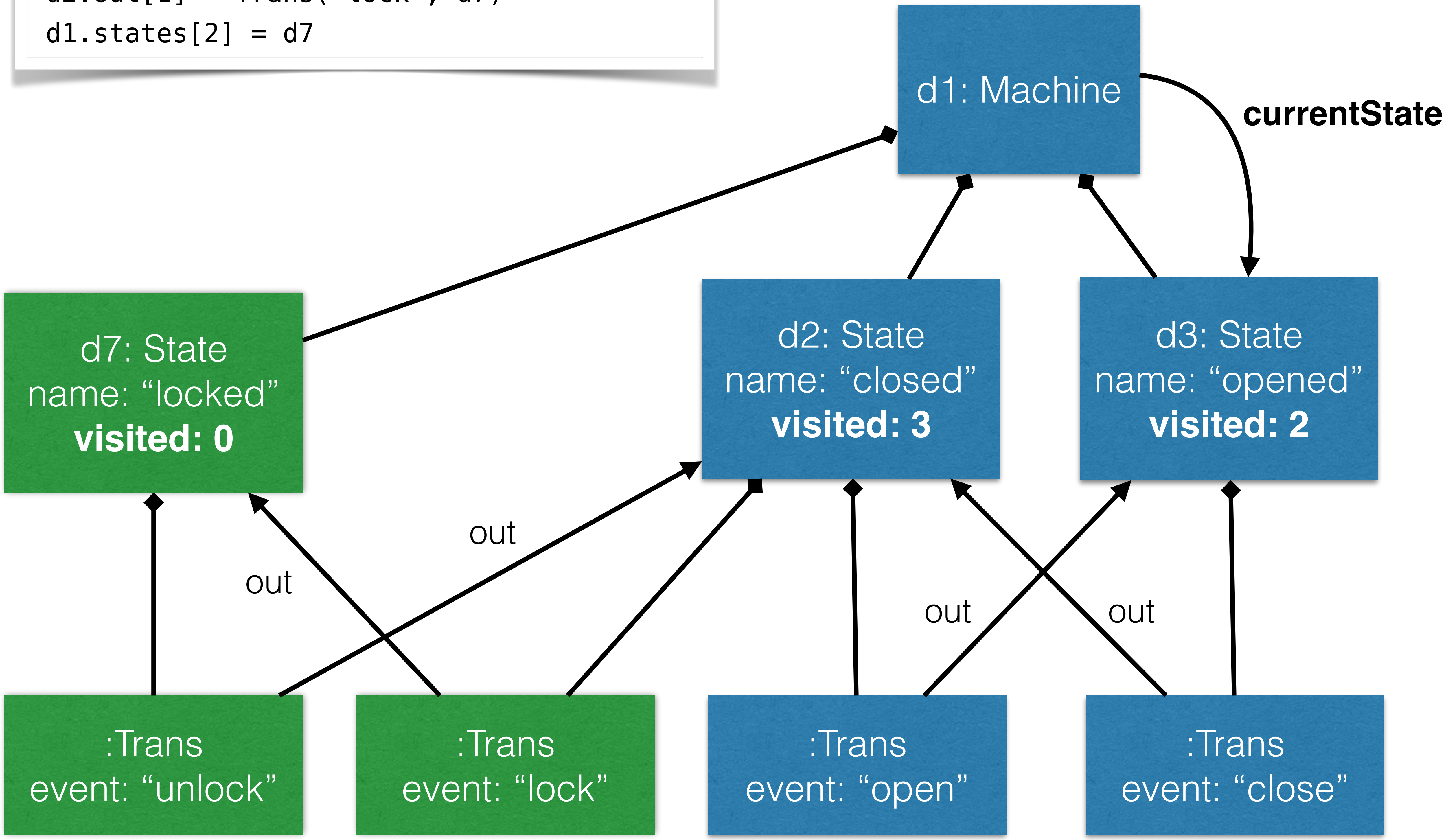
,

```
machine doors d4
  state closed d5
    open => opened u3
    lock => locked u4

  state opened d6
    close => closed u5

  state locked d7
    unlock => closed u6

end
```

) =

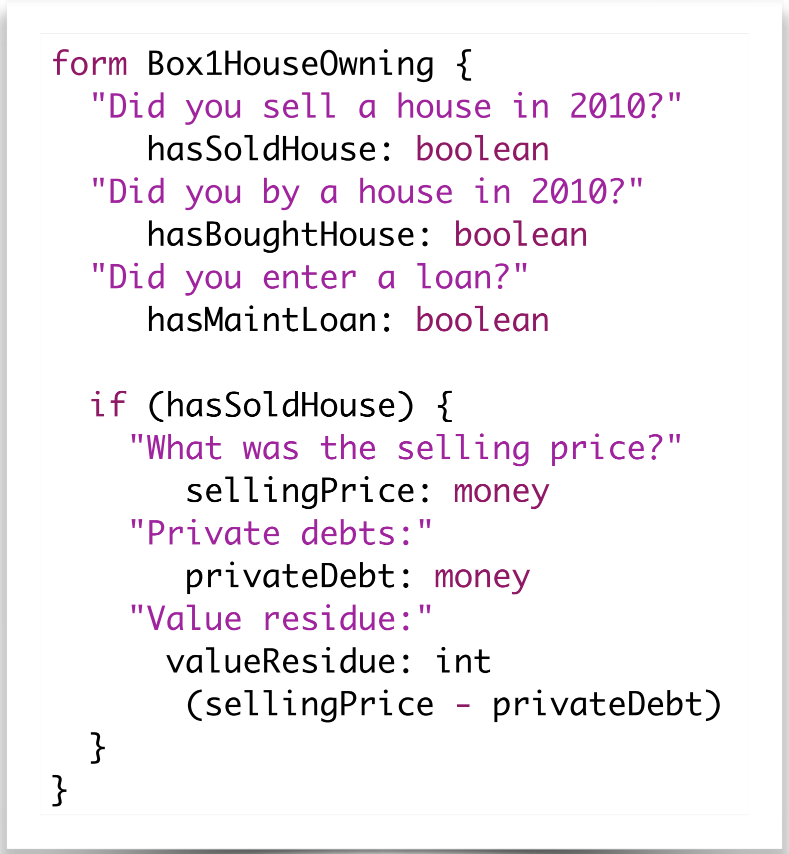
```
create State d7
d7 = State("locked",[Trans("unlock",d2)])
d2.out[1] = Trans("lock", d7)
d1.states[2] = d7
```

```
create State d7
d7 = State("locked",[Trans("unlock",d2)])
d2.out[1] = Trans("lock", d7)
d1.states[2] = d7
```



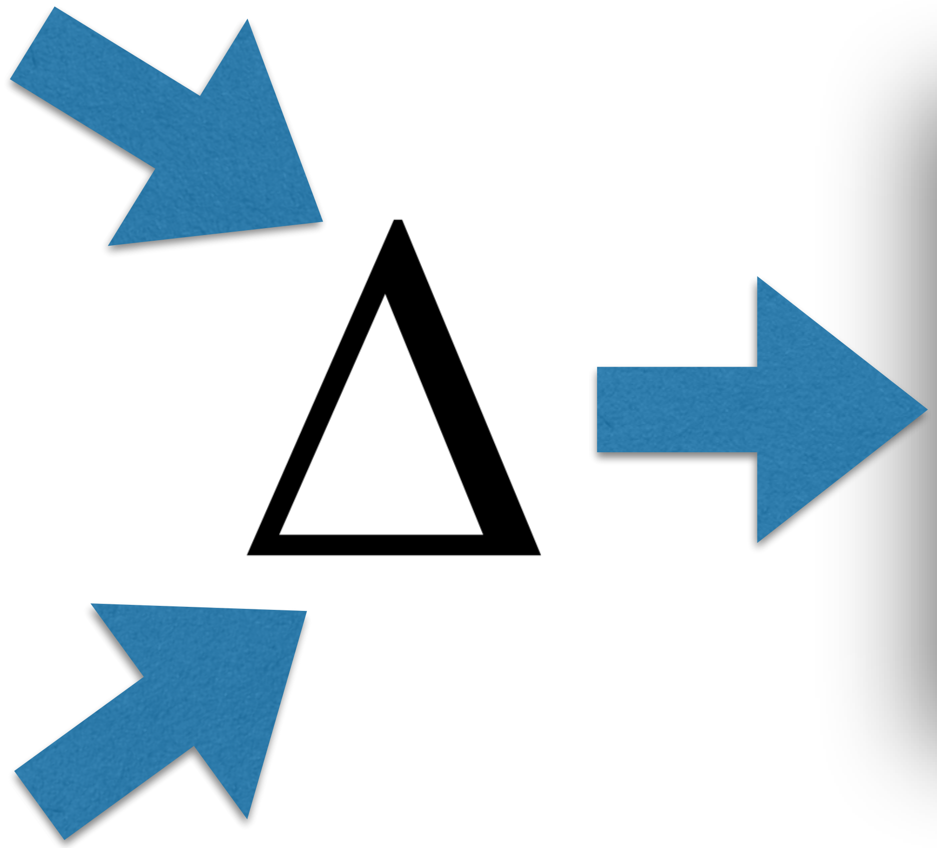
Potential

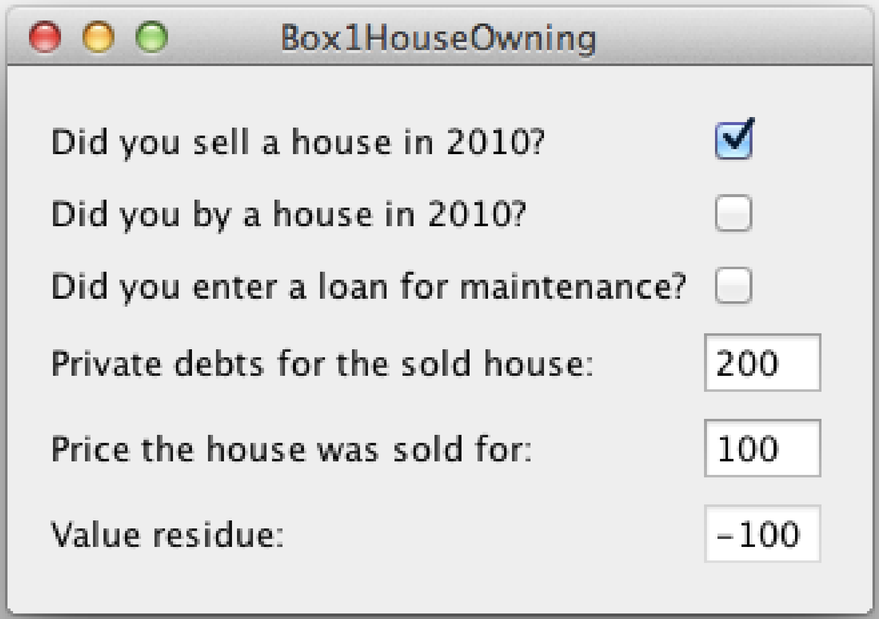
- Time travel (undo, the inevitable slider)
- Time branching (what-if scenarios)
- Merging (!?!?@!@!?@#@#%%%)
- Persistence (EventStores!)
- Versioning



```
form Box1HouseOwning {
  "Did you sell a house in 2010?"
     hasSoldHouse: boolean
  "Did you by a house in 2010?"
     hasBoughtHouse: boolean
  "Did you enter a loan?"
     hasMaintLoan: boolean

  if (hasSoldHouse) {
    "What was the selling price?"
       sellingPrice: money
    "Private debts:"
       privateDebt: money
    "Value residue:"
      valueResidue: int
       (sellingPrice - privateDebt)
  }
}
```

Code change

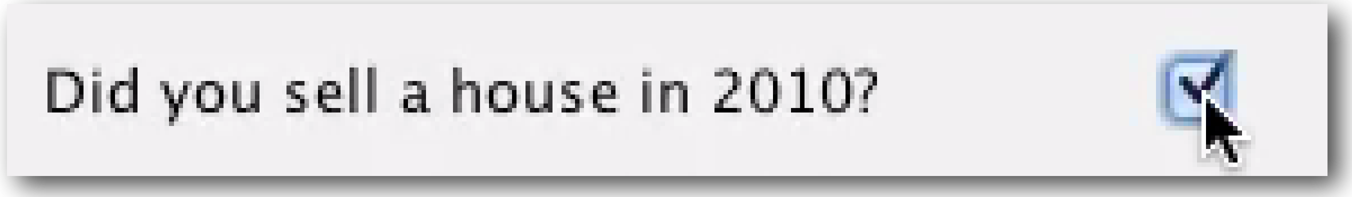




Box1HouseOwning

Did you sell a house in 2010?	✓
Did you by a house in 2010?	☐
Did you enter a loan for maintenance?	☐
Private debts for the sold house:	200
Price the house was sold for:	100
Value residue:	-100

User event



Did you sell a house in 2010? ✓

I have my live DSL environments (?)

- Generic “semantic” diff
- *Dynamically* apply *static* delta
- Decoupling of front-end and back-end
- Editing the program ~ interacting with the program
- Deltas!!!

Discussion

- Too good to be true? (Seems to work well)
- Nature of relation static model / runtime model?
  - (Inverted lens?)
- How general?
- Towards delta-oriented languages and systems