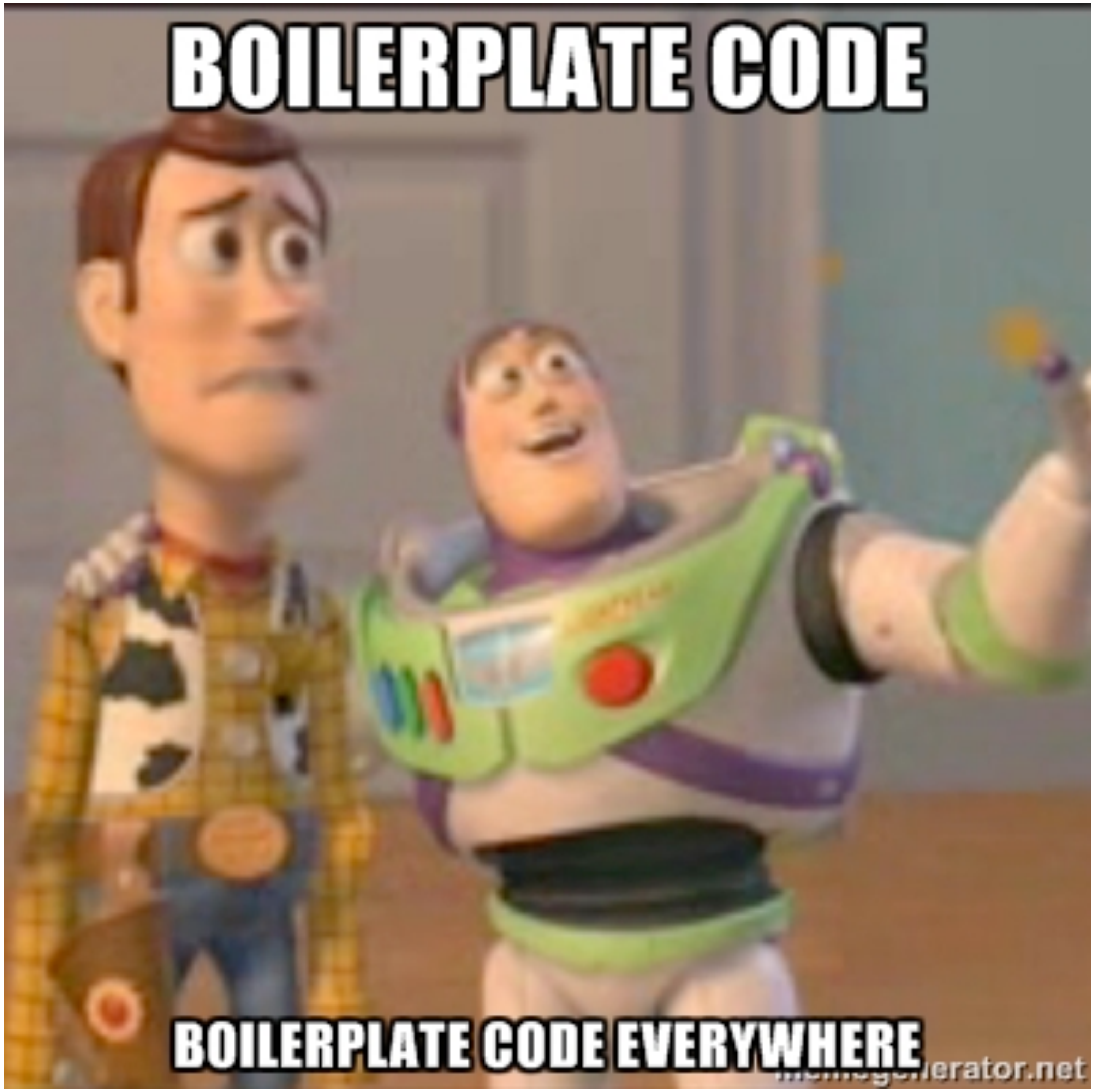
Domain-Specific Languages

Tijs van der Storm
(storm@cwi.nl / @tvdstorm)

CWI
Centrum Wiskunde & Informatica

University of Amsterdam

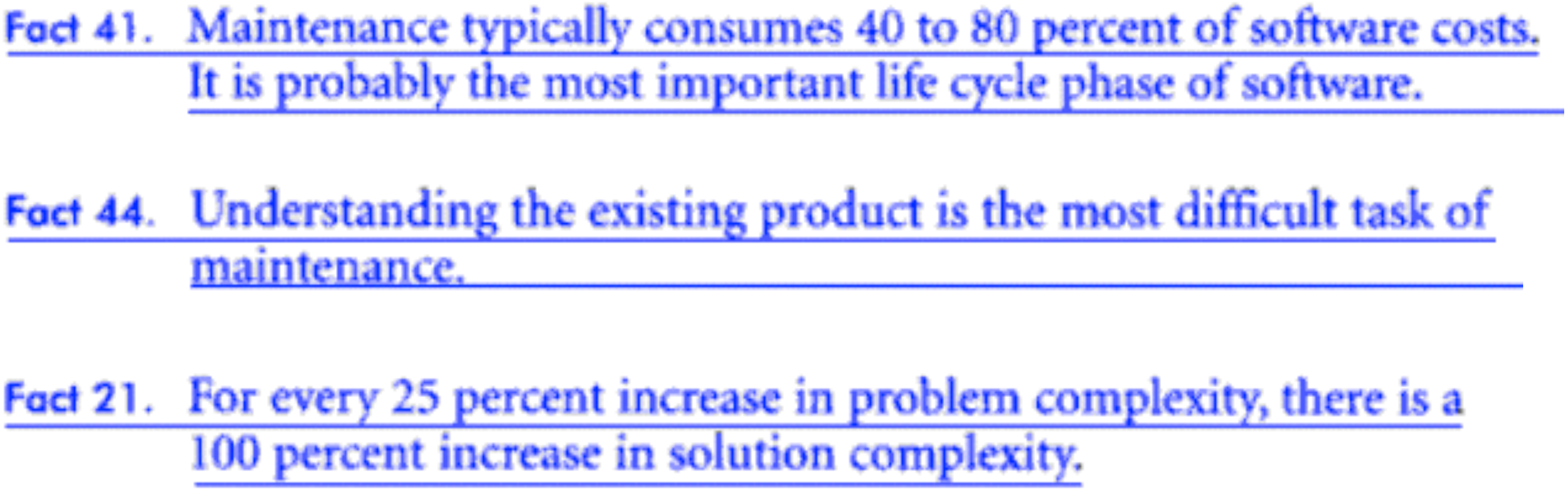
Programming?


BOILERPLATE CODE
BOILERPLATE CODE EVERYWHERE
memegenerator.net

A programming language is low level when its programs require attention to the irrelevant

Alan J. Perlis, Epigrams on Programming, *ACM SIGPLAN Notices* 17 (9), September 1982, pp. 7–13

Some facts



Fact 41. Maintenance typically consumes 40 to 80 percent of software costs. It is probably the most important life cycle phase of software.

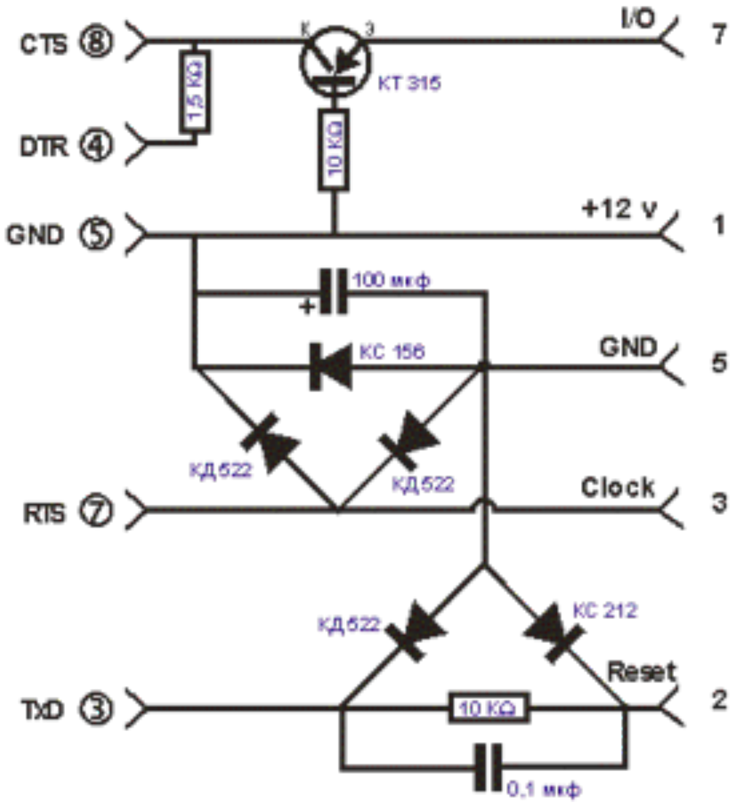
Fact 44. Understanding the existing product is the most difficult task of maintenance.

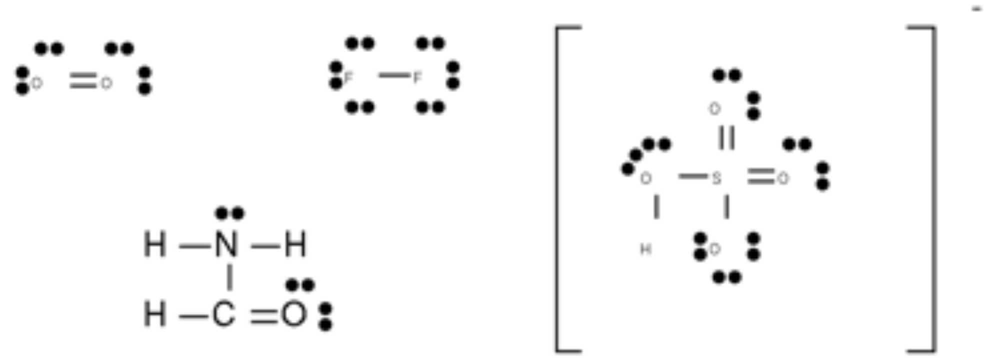
Fact 21. For every 25 percent increase in problem complexity, there is a 100 percent increase in solution complexity.

Domain Specific Languages!

Robert Glass, *Facts and fallacies of Software Engineering*, Addison-Wesley 2003

Domain specific languages

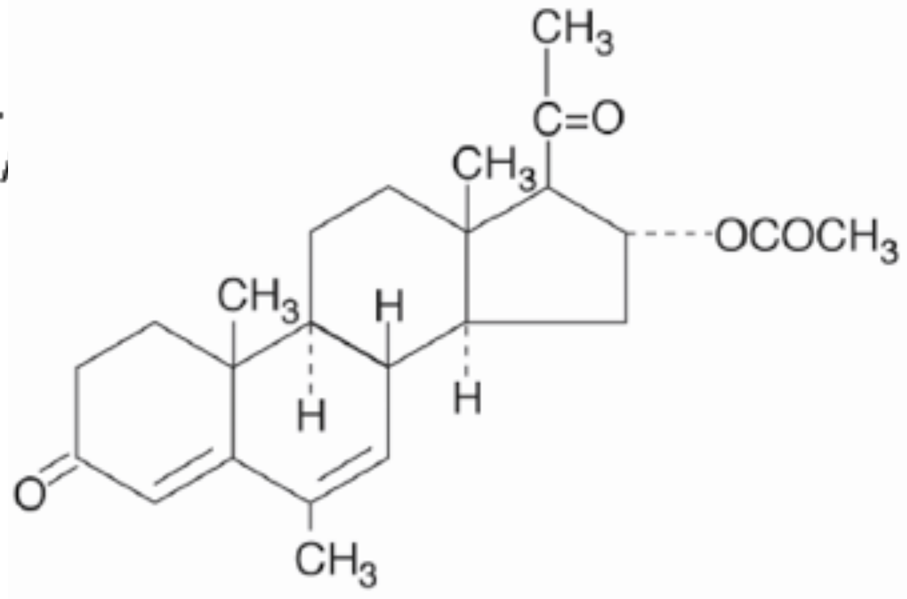




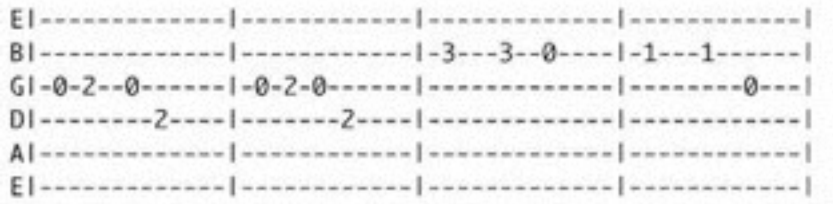
$$c_t = S_t N(h) - X e^{-rt} N(h - \sigma\sqrt{\tau})$$

where

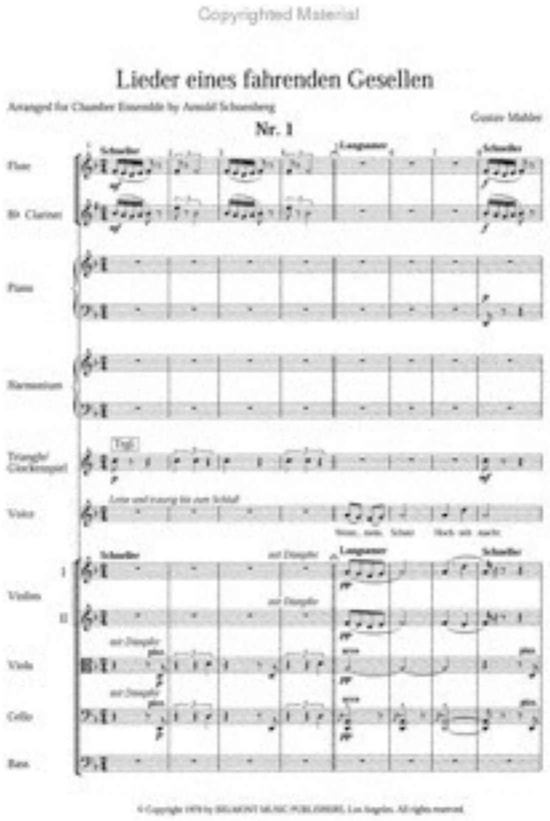
$$h = \{\ln(\frac{S}{X}) + r\tau + \frac{\sigma^2\tau}{2}\}$$

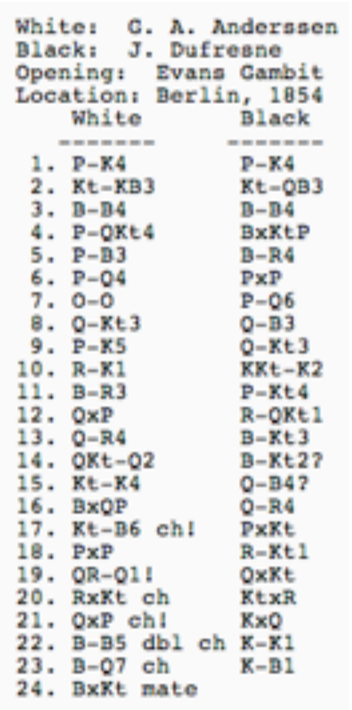


Domain specific languages

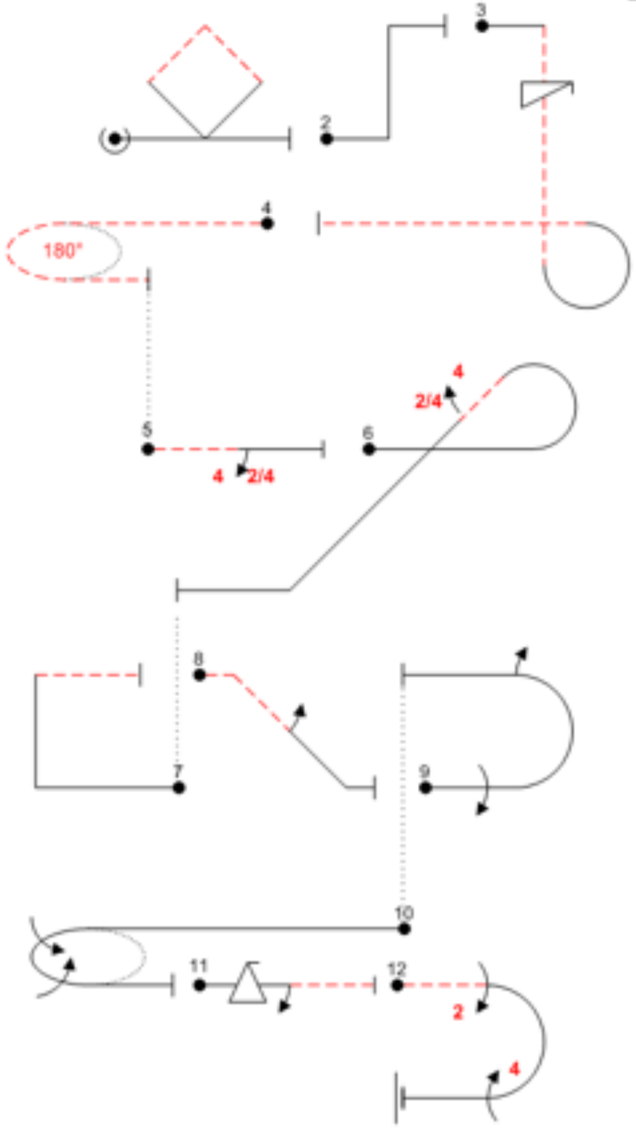


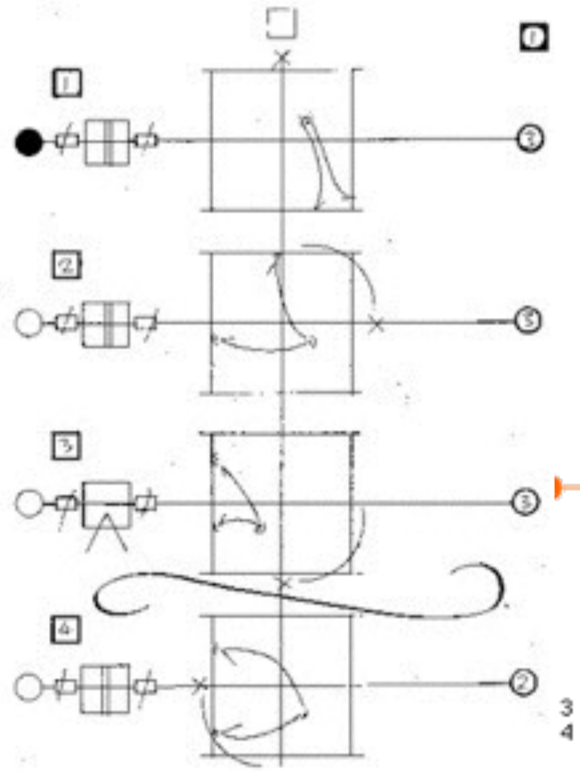
```
E|-------------|------------|-------------|------------|
B|-------------|------------|-3---3--0----|-1---1------|
G|-0-2--0------|-0-2-0------|-------------|--------0---|
D|--------2----|-------2----|-------------|------------|
A|-------------|------------|-------------|------------|
E|-------------|------------|-------------|------------|
```

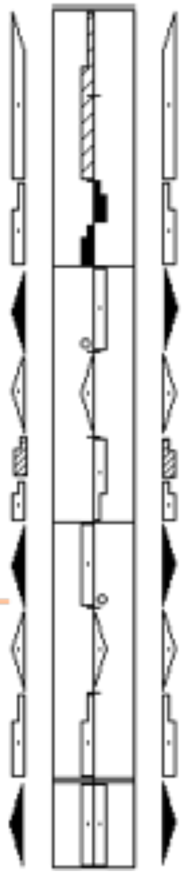

Copyrighted Material
Lieder eines fahrenden Gesellen
Arranged for Chamber Ensemble by Arnold Schoenberg
Gustav Mahler
Nr. 1
Schneller
Langsamer
Flute
B♭ Clarinet
Piano
Harmonium
Triangle/ Glockenspiel
Voice
Violins
I
II
Viola
Cello
Bass
Leise und traurig bis zum Schluß
mit Dämpfer
Copyright 1979 by BELMONT MUSIC PUBLISHERS, Los Angeles. All rights reserved.



```
White:  G. A. Anderssen
Black:  J. Dufresne
Opening:  Evans Gambit
Location: Berlin, 1854
     White         Black
    -------       -------
  1. P-K4          P-K4
  2. Kt-KB3        Kt-QB3
  3. B-B4          B-B4
  4. P-QKt4        BxKtP
  5. P-B3          B-R4
  6. P-Q4          PxP
  7. O-O           P-Q6
  8. Q-Kt3         Q-B3
  9. P-K5          Q-Kt3
 10. R-K1          KKt-K2
 11. B-R3          P-Kt4
 12. QxP           R-QKt1
 13. Q-R4          B-Kt3
 14. QKt-Q2        B-Kt2?
 15. Kt-K4         Q-B4?
 16. BxQP          Q-R4
 17. Kt-B6 ch!     PxKt
 18. PxP           R-Kt1
 19. QR-Q1!        QxKt
 20. RxKt ch       KtxR
 21. QxP ch!       KxQ
 22. B-B5 dbl ch   K-K1
 23. B-Q7 ch       K-B1
 24. BxKt mate
```







Observations

- Special purpose
- Restricted
- Concise
- Expert usage
- Formalized
- Textual or graphic or combination

General purpose languages (GPLs)

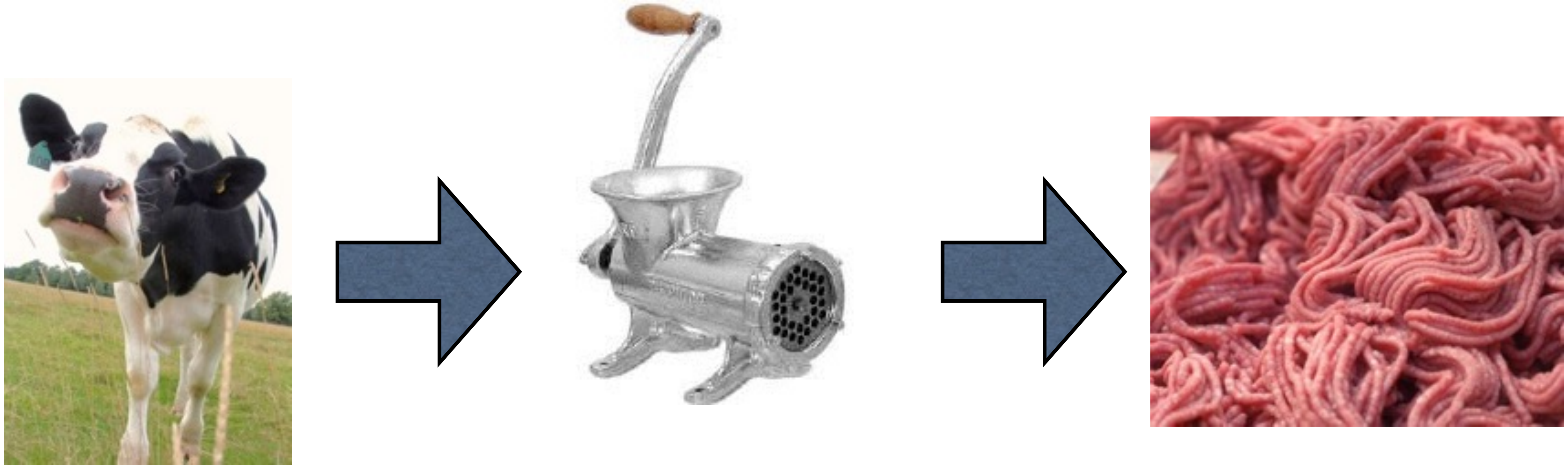
DSLs

Programming

```
.text:131411EF                         ; FUNCTION CHUNK AT .text:13141239 SIZE 000
.text:131411EF                         ; FUNCTION CHUNK AT .vmp0:13143000 SIZE 000
.text:131411EF                         ; FUNCTION CHUNK AT .vmp0:13143522 SIZE 000
.text:131411EF                         ; FUNCTION CHUNK AT .vmp0:13145C99 SIZE 000
.text:131411EF
.text:131411EF 68 F1 4F 58 FF                          push    0FF584FF1h
.text:131411F4 E9 A0 4A 00 00                          jmp     loc_13145C99
.text:131411F4                         start           endp
.text:131411F4
.text:131411F4                         ; ---------------------------------------
.text:131411F9 64 69 31                byte_131411F9   db 64h, 69h, 31h        ; D
.text:131411FC 60 16 42 75 9A C4+                      dd 75421660h, 0CACDC49Ah, 7
.text:131411FC CD CA FA 7B 9D 7B+                      dd 6EEE544Dh, 10D69610h, 0F
.text:131411FC D5 D4 48 2B 85 4B+                      dd 0EB6323E4h, 82858465h, 0
.text:13141238                         ; ---------------------------------------
.text:13141238 58                                      pop     eax
.text:13141239                         ; START OF FUNCTION CHUNK FOR start
.text:13141239
.text:13141239                         loc_13141239:                           ; C
.text:13141239 52                                      push    edx
.text:1314123A 55                                      push    ebp
.text:1314123B 53                                      push    ebx
.text:1314123C 51                                      push    ecx
.text:1314123D 9C                                      pushf
.text:1314123E 57                                      push    edi
.text:1314123F 50                                      push    eax
.text:13141240 56                                      push    esi
.text:13141241 51                                      push    ecx
.text:13141242 68 00 00 00 00                          push    0
.text:13141247 8B 74 24 28                             mov     esi, [esp+2Ch+var_4
.text:1314124B BF F9 11 14 13                          mov     edi, offset byte_13
.text:13141250
.text:13141250                         loc_13141250:                           ; C
.text:13141250 89 F3                                   mov     ebx, esi
.text:13141252 03 34 24                                add     esi, [esp+2Ch+var_2
.text:13141255
.text:13141255                         loc_13141255:                           ; C
.text:13141255                                                                 ; s
```

Domain Programmer Code

Programming



Domain

Programmer

Code

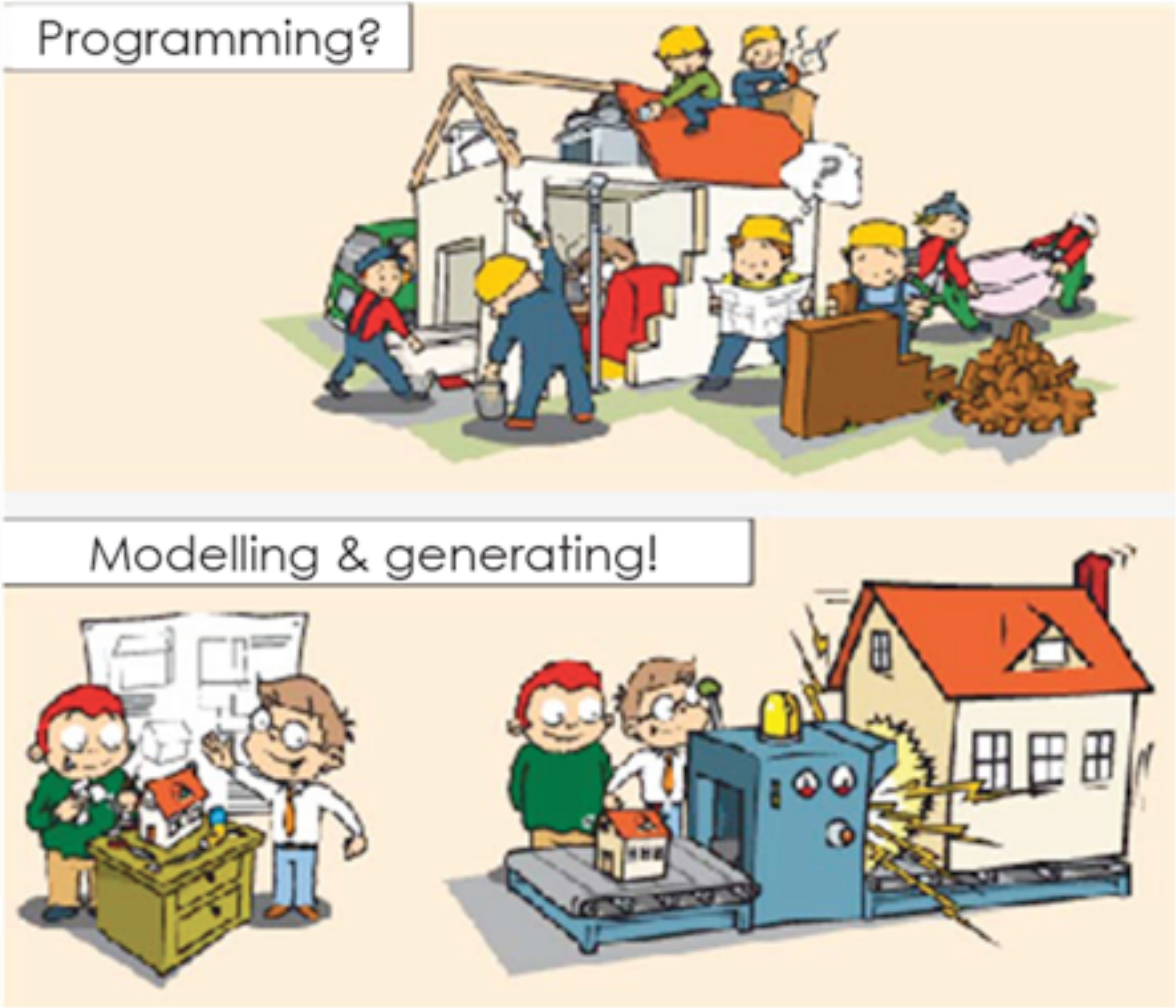
Programming is “lossy”

- encoding
- obfuscating
- encrypting
- dispersing
- tangling
- distorting

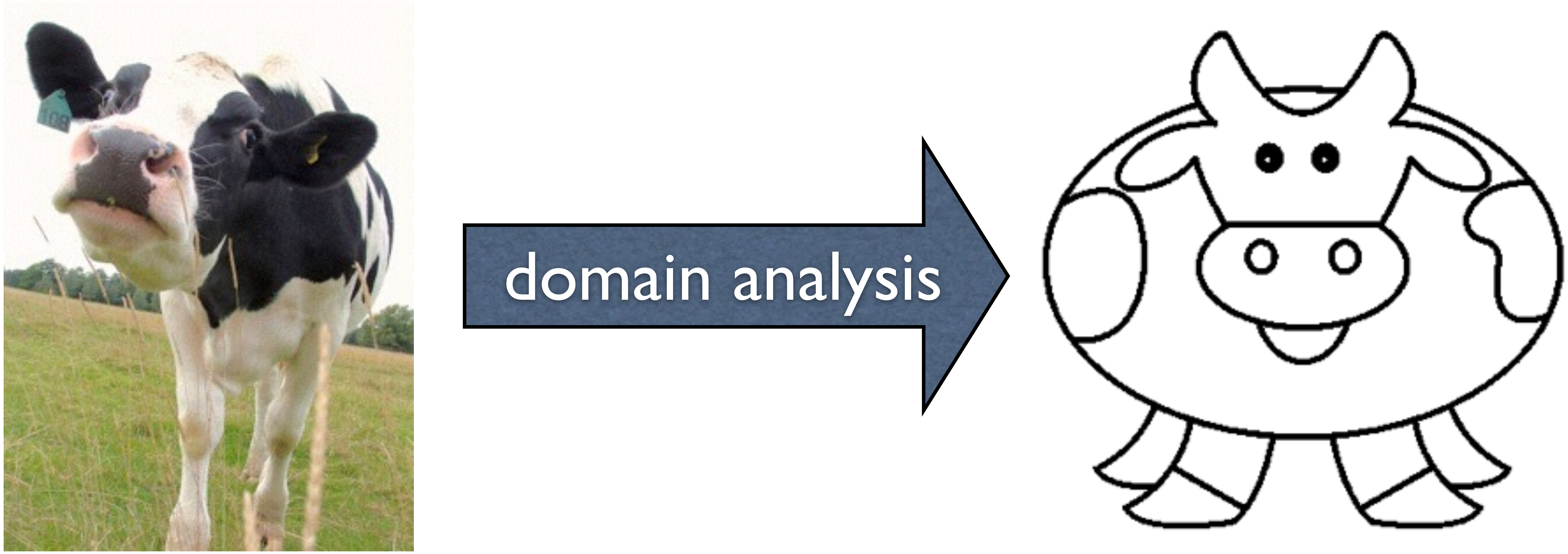
Cognitive distance

The problem

- a lot of code,
- low level code,
- characterized by lack of abstraction
- encoding domain knowledge
- and encoding design knowledge


Programming?
Modelling & generating!

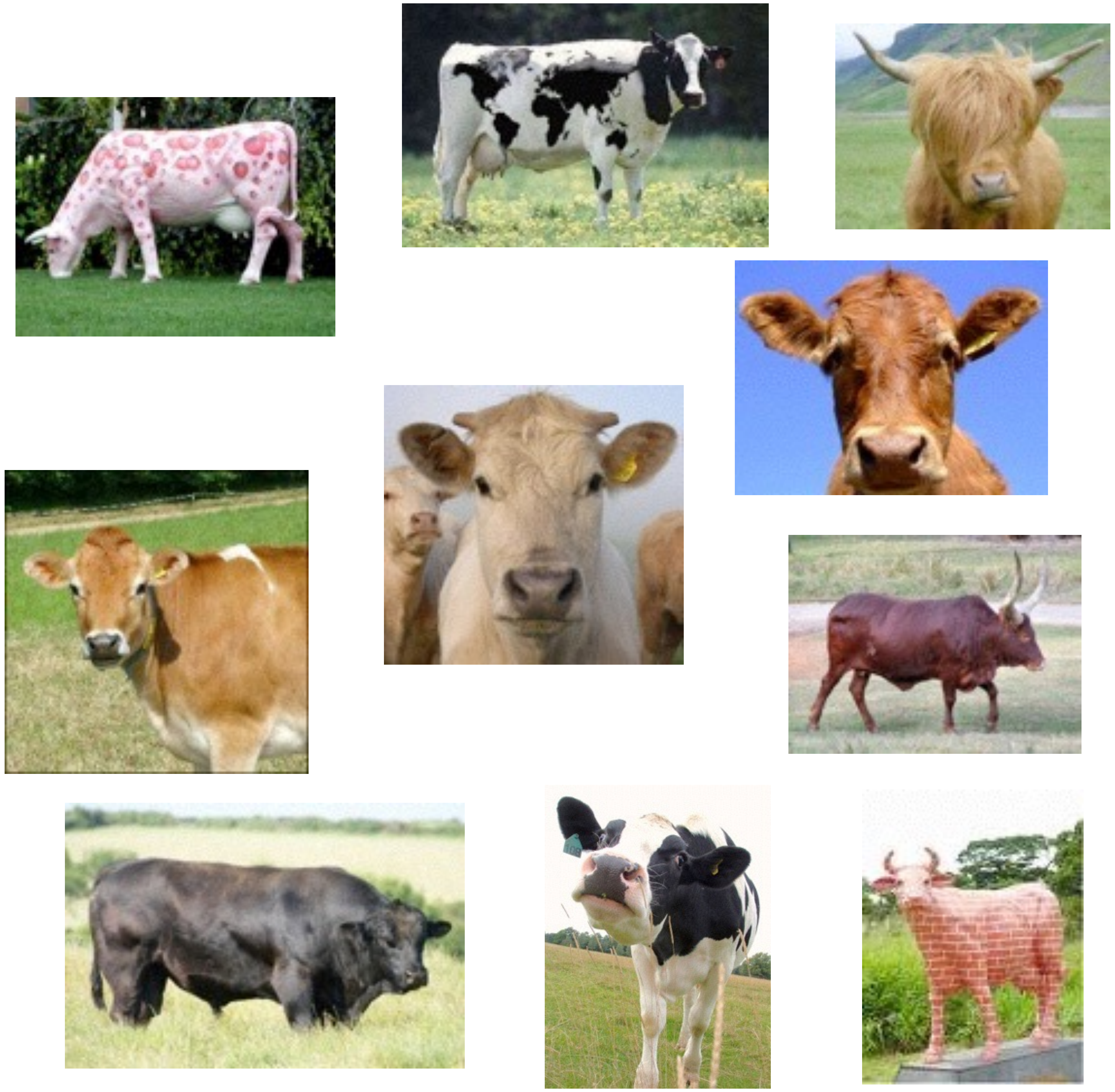
Modeling the domain

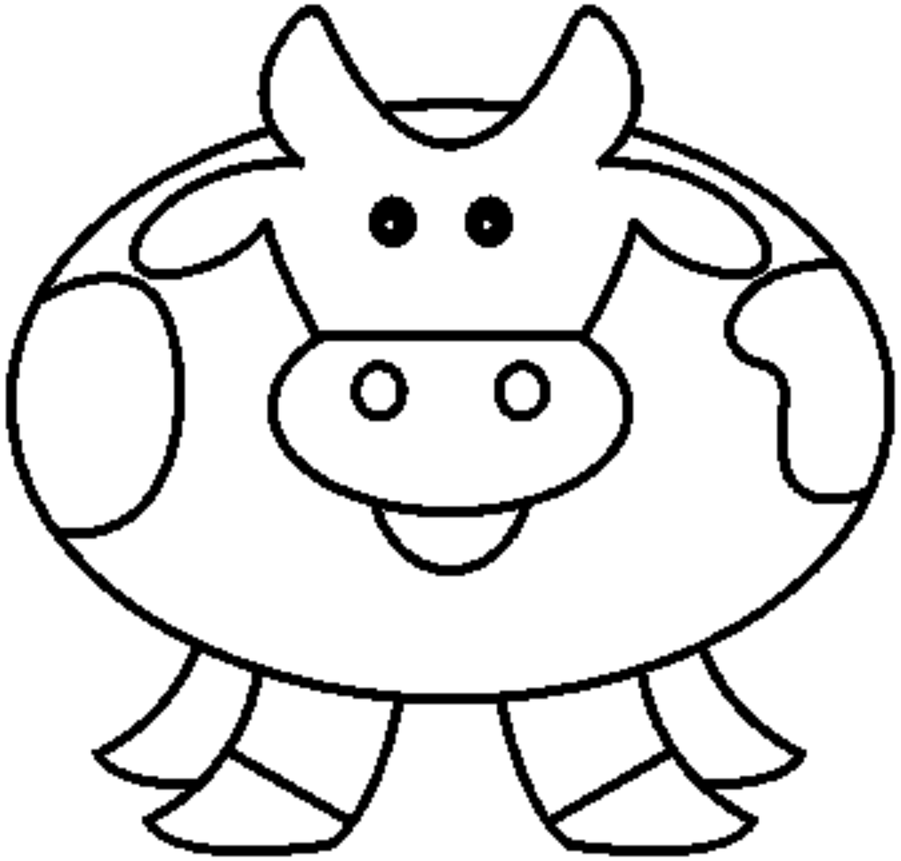


domain analysis

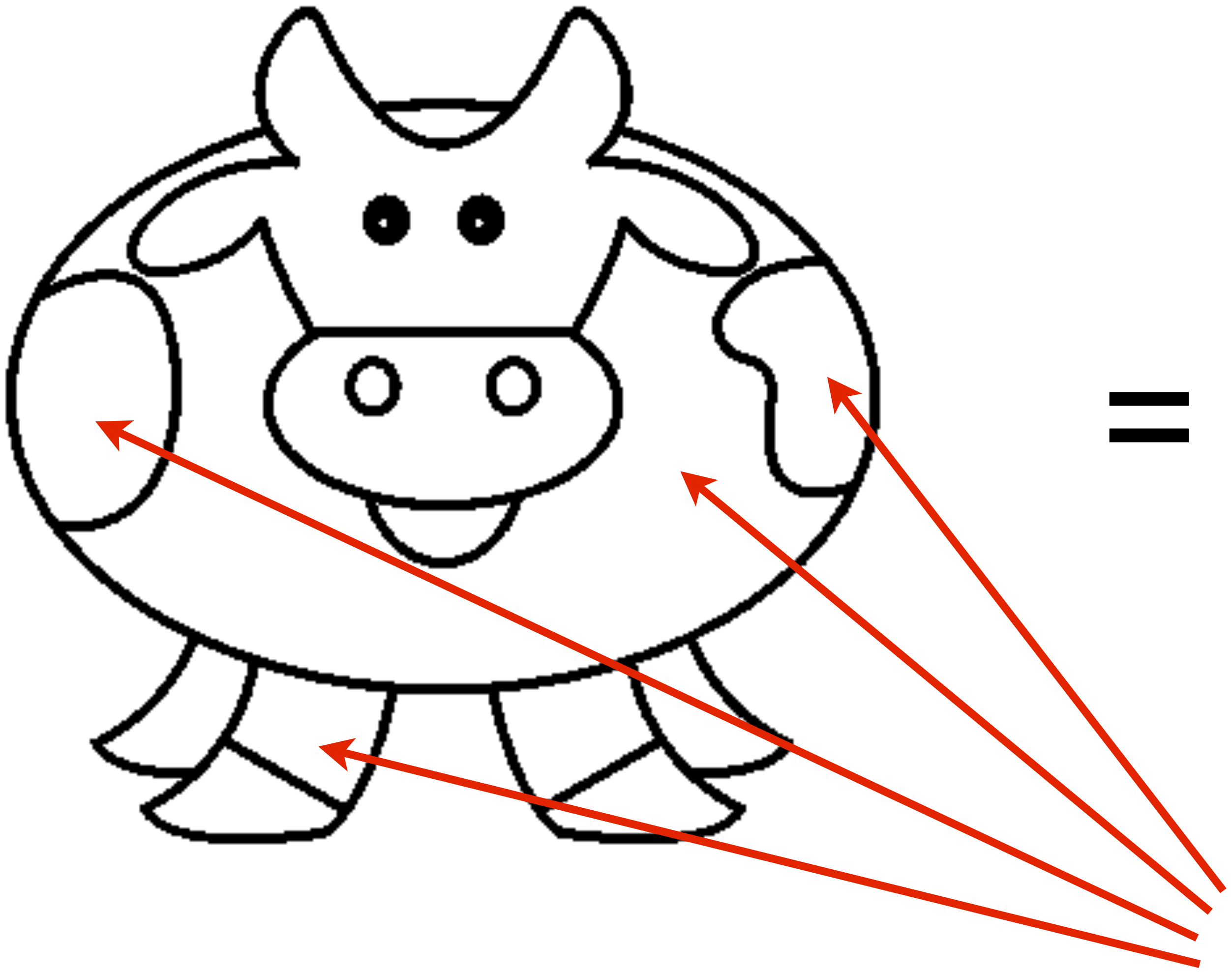
Ceci n’est pas une vache

System families





Domain Specific Language

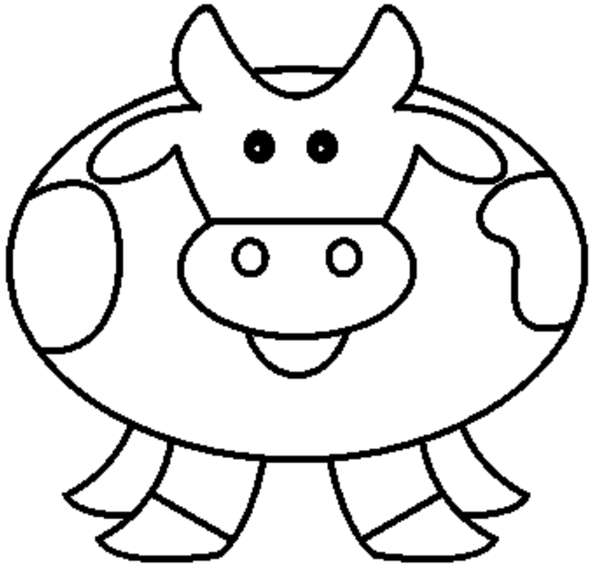


variation points

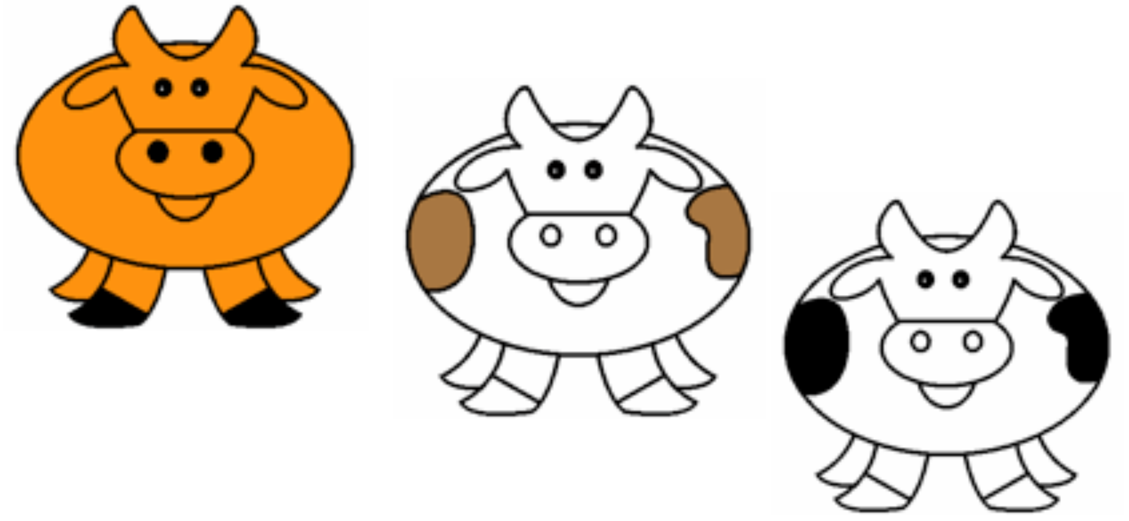
=

formalized notation capturing "Cows"

Domain Specific Languages

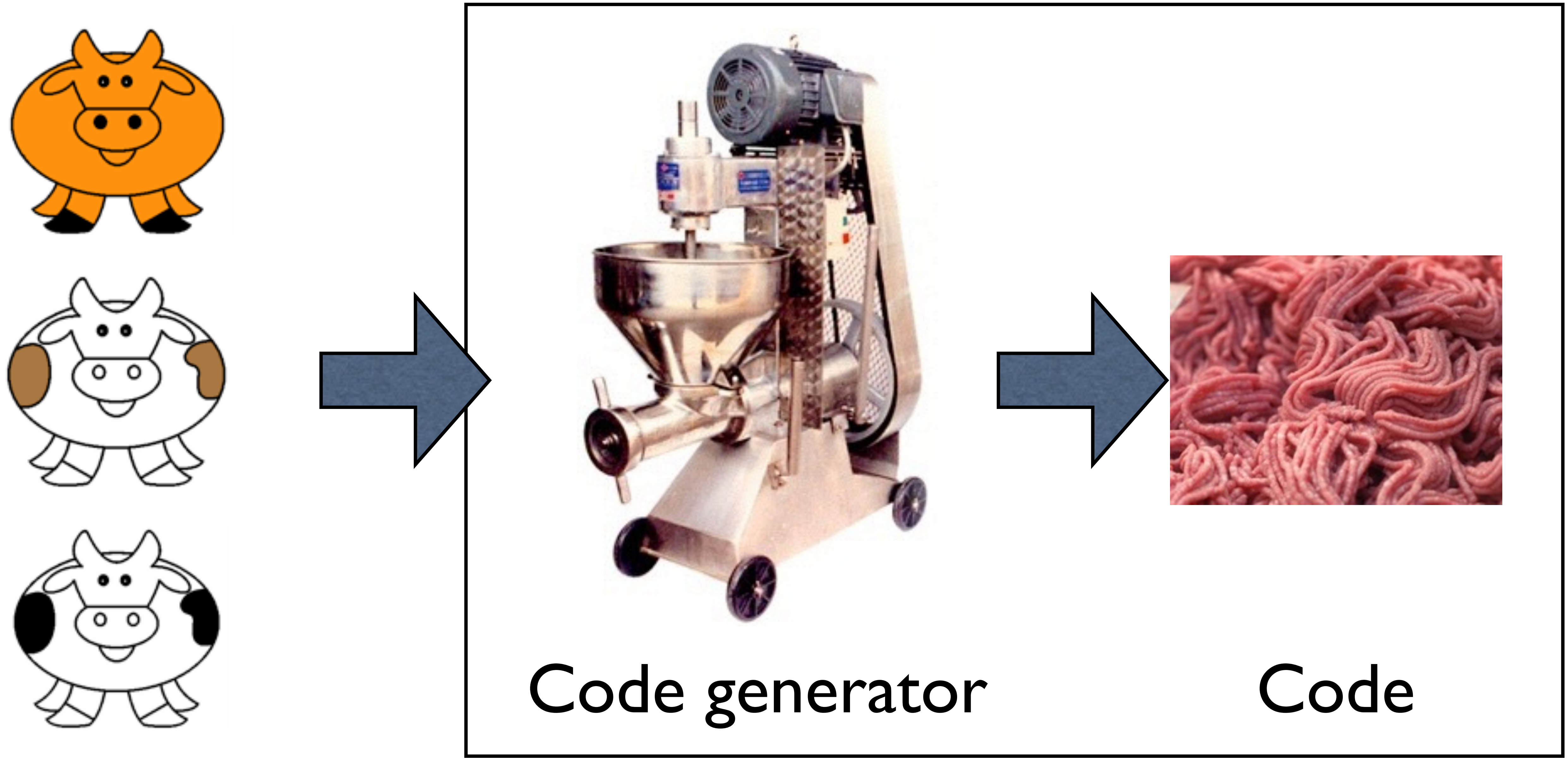


= grammar, template, metamodel



= sentence, instance, model

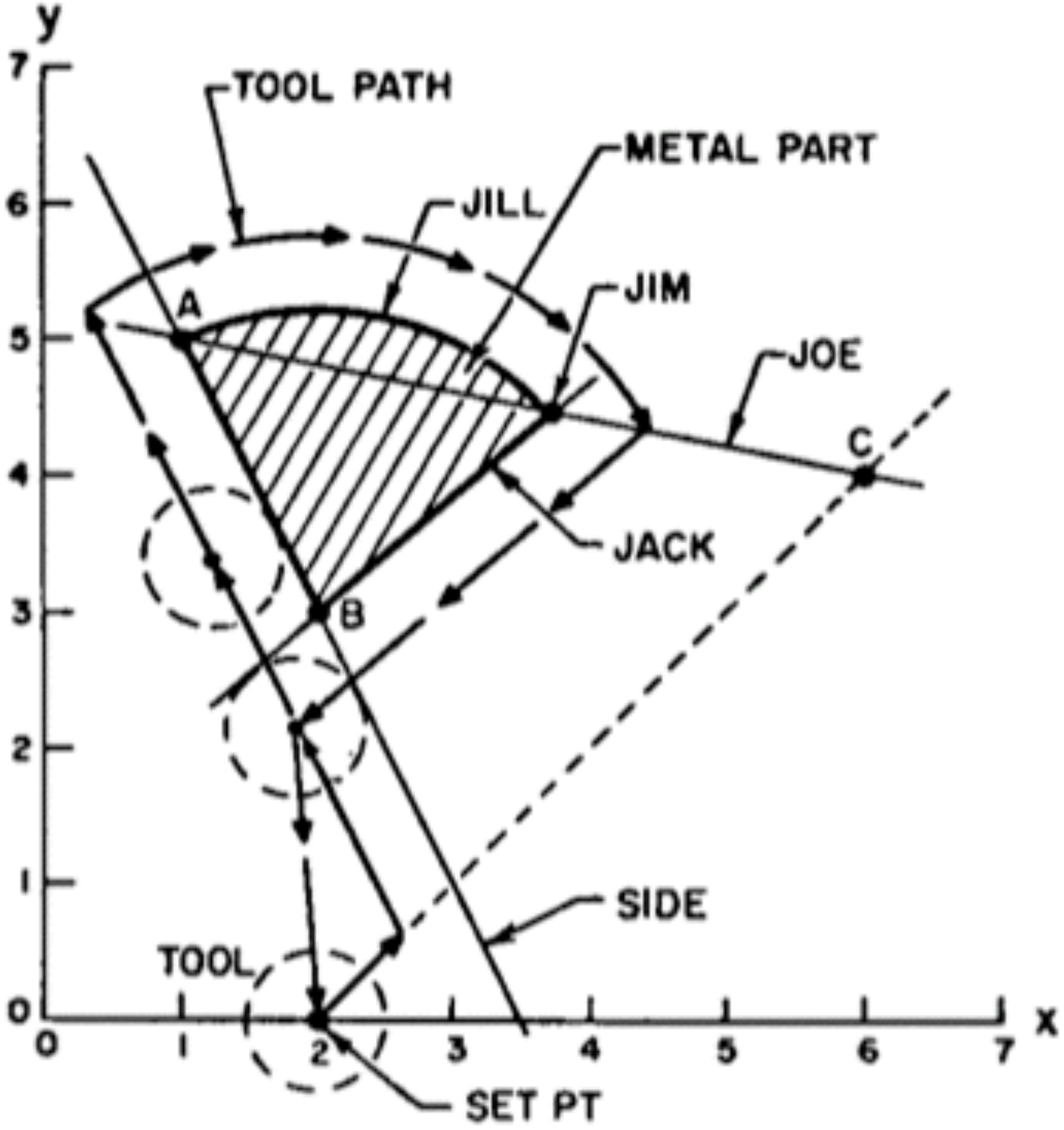
Code generation



Code generator

Code

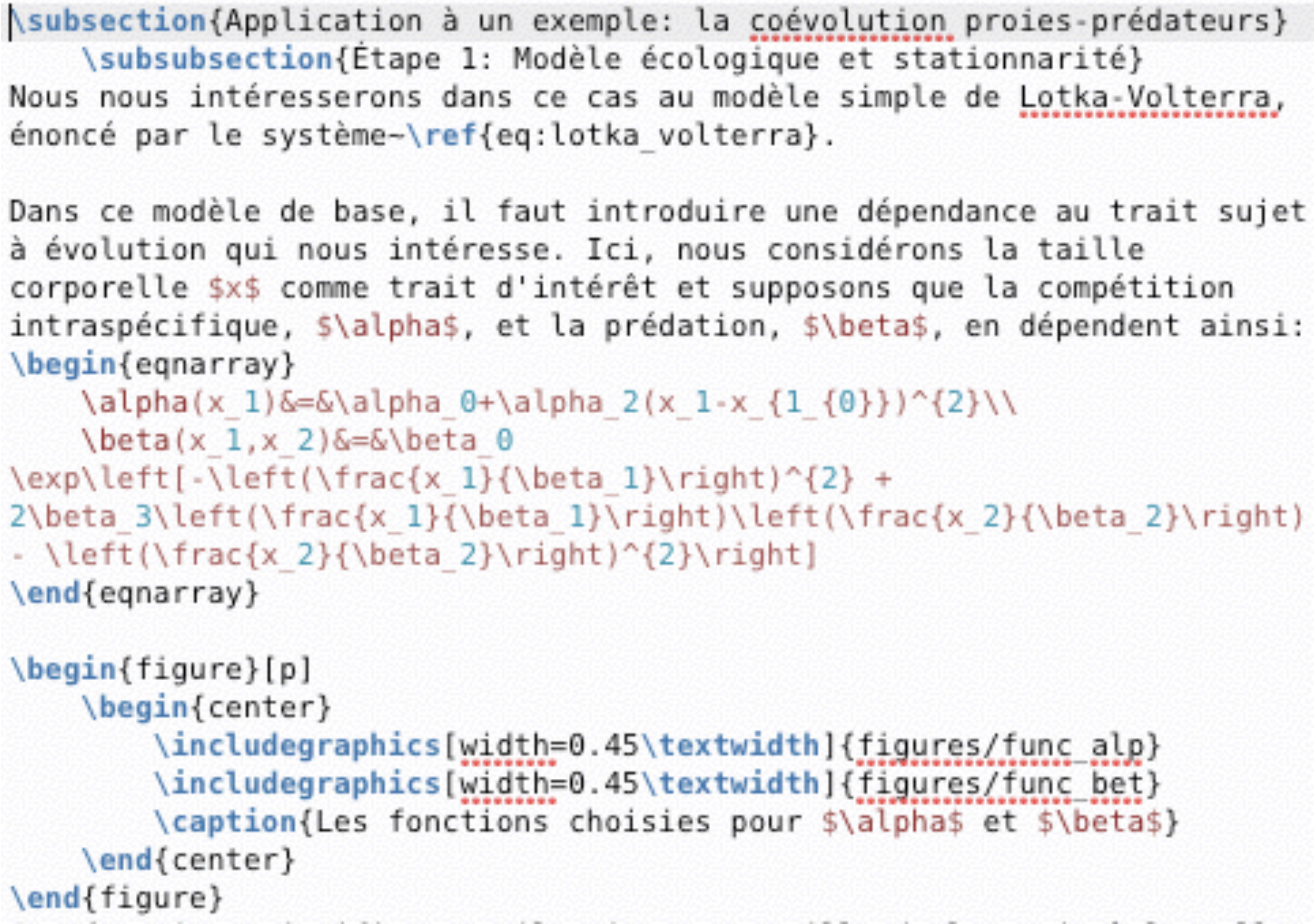
APT: numerical control



```
     A  =  POINT / 1, 5
     B  =  POINT / 2, 3
     C  =  POINT / 6, 4
           TL DIA / +1.0, INCH
           FEDRAT / 30, IPM
SET PT  =  FROM, POINT / 2, 0
           IN DIR, POINT / C
  SIDE  =  GO TO, LINE / THRU, A, AND, B
           WITH, TL LFT, GO LFT, ALONG / SIDE
  JILL  =  GO RGT, ALONG, CIRCLE / WITH, CTR AT, B, THRU, A
   JOE  =  LINE / THRU, A, AND, C
   JIM  =  POINT / X LARGE, INT OF, JOE, WITH, JILL
  JACK  =  LINE / THRU, JIM, AND, B
           GO RGT, ALONG / JACK, UNTIL, TOOL, PAST, SIDE
           GO TO / SET PT
           STOP, END, FINI
```

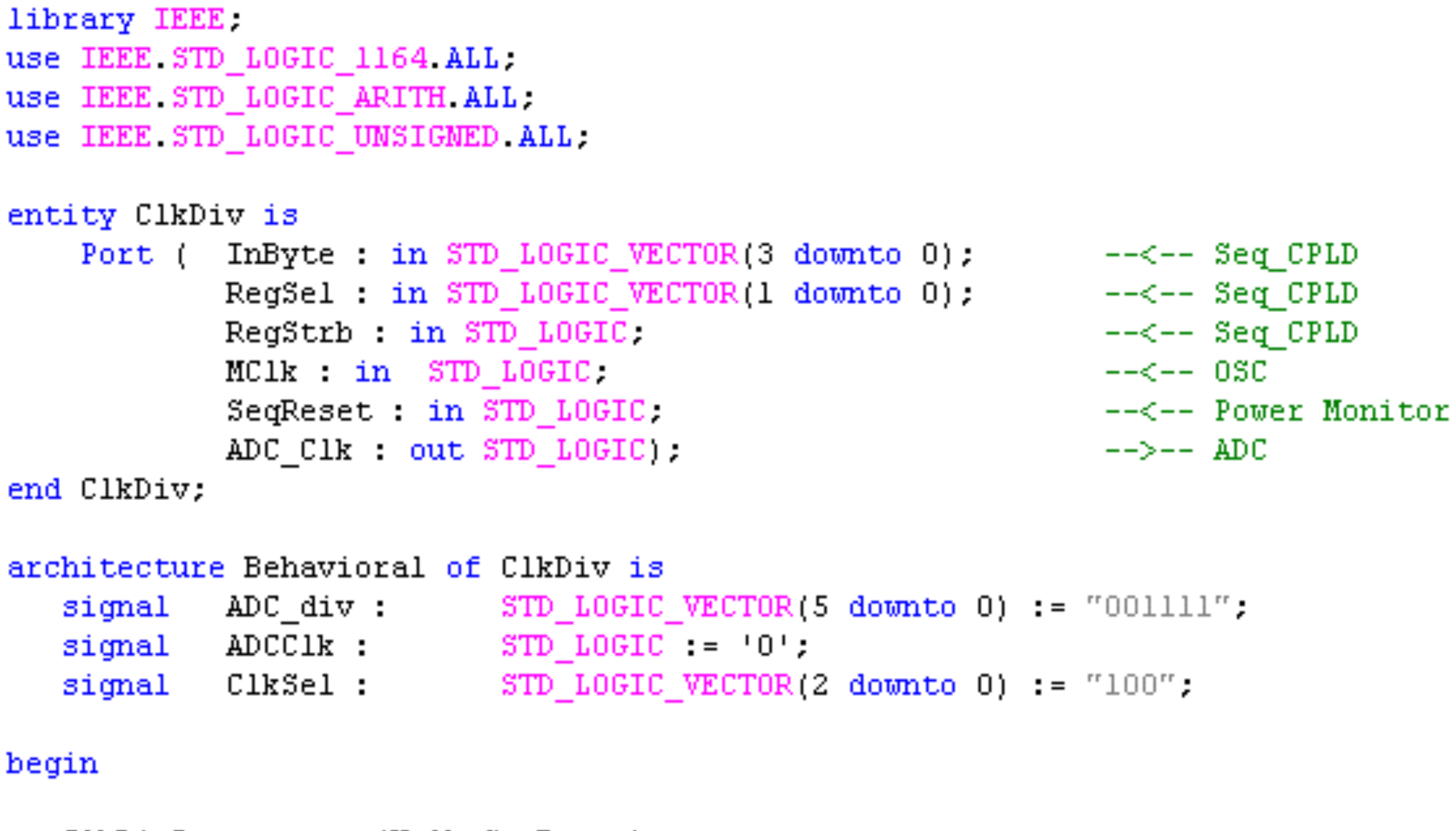
from
the ’50s (!)

LaTeX: document preparation



```
\subsection{Application à un exemple: la coévolution proies-prédateurs}
    \subsubsection{Étape 1: Modèle écologique et stationnarité}
Nous nous intéresserons dans ce cas au modèle simple de Lotka-Volterra,
énoncé par le système~\ref{eq:lotka_volterra}.

Dans ce modèle de base, il faut introduire une dépendance au trait sujet
à évolution qui nous intéresse. Ici, nous considérons la taille
corporelle $x$ comme trait d'intérêt et supposons que la compétition
intraspécifique, $\alpha$, et la prédation, $\beta$, en dépendent ainsi:
\begin{eqnarray}
    \alpha(x_1)&=&\alpha_0+\alpha_2(x_1-x_{1_{0}})^{2}\\
    \beta(x_1,x_2)&=&\beta_0
\exp\left[-\left(\frac{x_1}{\beta_1}\right)^{2} +
2\beta_3\left(\frac{x_1}{\beta_1}\right)\left(\frac{x_2}{\beta_2}\right)
- \left(\frac{x_2}{\beta_2}\right)^{2}\right]
\end{eqnarray}

\begin{figure}[p]
    \begin{center}
        \includegraphics[width=0.45\textwidth]{figures/func_alp}
        \includegraphics[width=0.45\textwidth]{figures/func_bet}
        \caption{Les fonctions choisies pour $\alpha$ et $\beta$}
    \end{center}
\end{figure}
```

VHDL: hardware description



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ClkDiv is
    Port (  InByte : in STD_LOGIC_VECTOR(3 downto 0);         --<-- Seq_CPLD
            RegSel : in STD_LOGIC_VECTOR(1 downto 0);         --<-- Seq_CPLD
            RegStrb : in STD_LOGIC;                           --<-- Seq_CPLD
            MClk : in  STD_LOGIC;                             --<-- OSC
            SeqReset : in STD_LOGIC;                          --<-- Power Monitor
            ADC_Clk : out STD_LOGIC);                         -->-- ADC
end ClkDiv;

architecture Behavioral of ClkDiv is
   signal   ADC_div :      STD_LOGIC_VECTOR(5 downto 0) := "001111";
   signal   ADCClk :       STD_LOGIC := '0';
   signal   ClkSel :       STD_LOGIC_VECTOR(2 downto 0) := "100";

begin
```

Risla: financial products

Time to market went down from 3 months to 3 weeks.

Developed at CWI

```
product LOAN

declaration
  contract data
    PAMOUNT : amount                      %% Principal Amount
    STARTDATE : date                      %% Starting date
    MATURDATE : date                      %% Maturity data
    INTRATE : int-rate                    %% Interest rate
    RDMLIST := [] : cashflow-list         %% List of redemptions.

  information
    PAF : cashflow-list                   %% Principal Amount Flow
    IAF : cashflow-list                   %% Interest Amount Flow

  registration
    %% Register one redemption.
    RDM(AMOUNT : amount, DATE : date)
```

QL

```
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 for maintenance?" hasMaintLoan: boolean
  if (hasSoldHouse) {
    "Private debts for the sold house:" privateDebt: money
    "Price the house was sold for:" sellingPrice: money
    "Value residue:" valueResidue = sellingPrice - privateDebt
  }
}
```

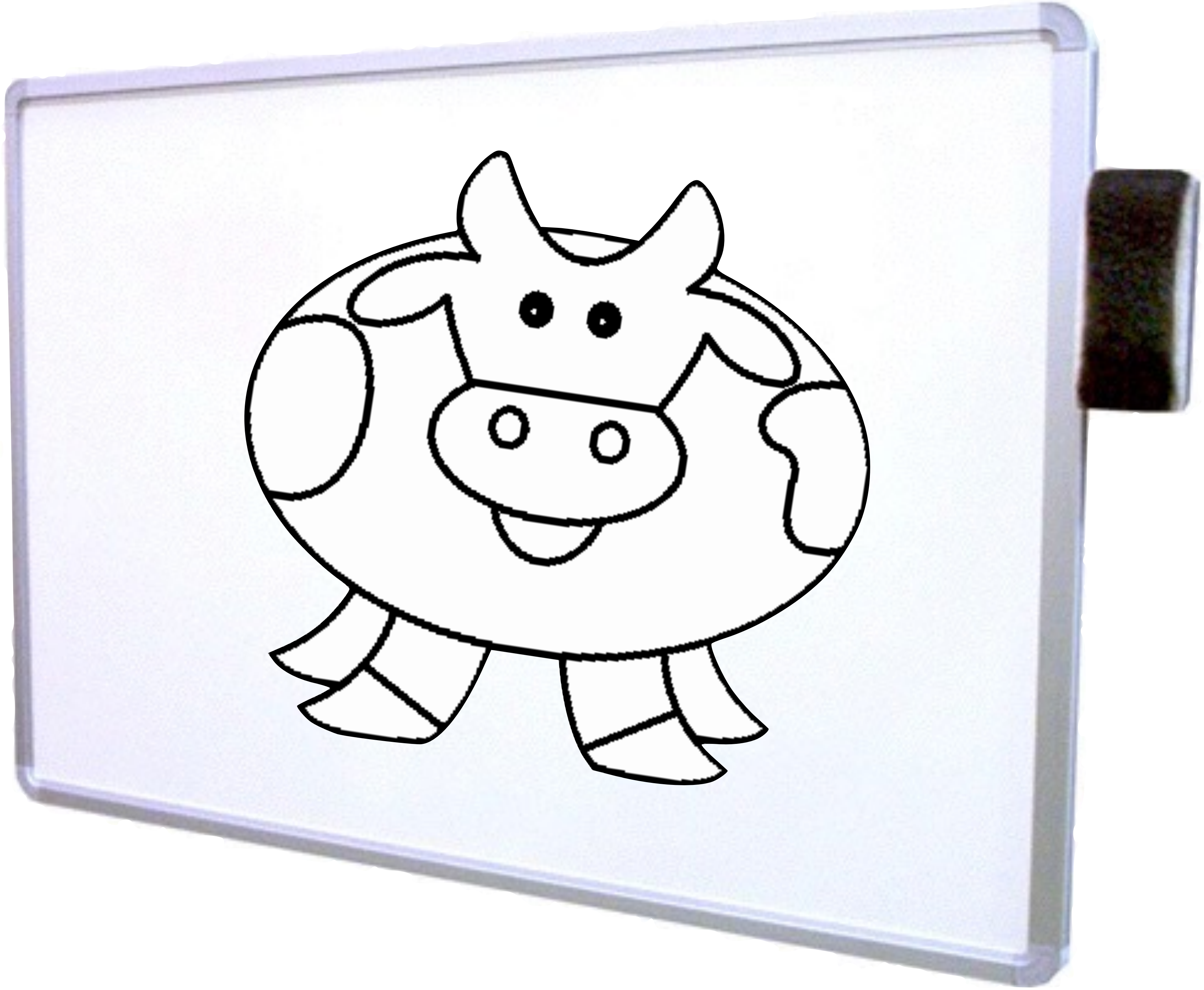
Other examples

- Make: software building
- Dot: graph visualization
- SQL: relational querying
- SWUL: Swing GUIs
- HTML: hypertext
- CLOPS: commandline options
- GNUPlot: plotting
- R: statistics
- CML: kernel config
- Lex: lexical scanning
- Excel: spreadheets
- Rascal: meta-programming
- ...

Domain-specific languages

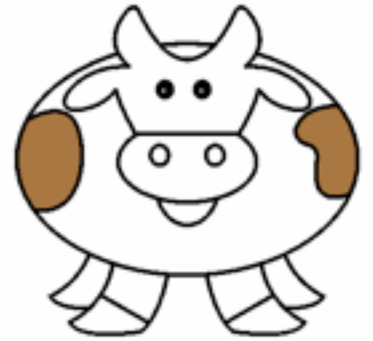
- Better languages, for specific domains
- Capture families of systems
- Higher level of abstraction
- Focus on “what” vs “how”
- Reuse designs, not just code
- Language workbenches (e.g., Rascal)

DSL Implementation

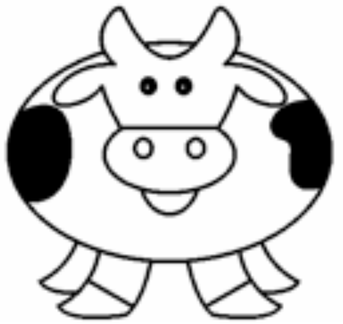


DSL Code

```
cow
  spots false
  color orange
end
```



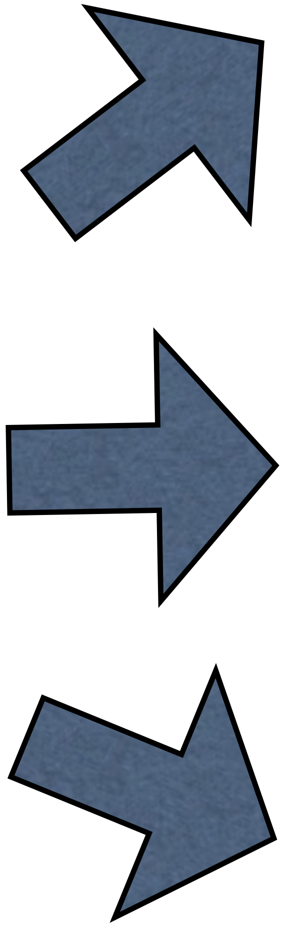
```
cow
  spots true
  color brown
end
```



```
cow
  spots true
  color black
end
```

Internal DSLs

```
cow
  spots false
  color orange
end
```



```
new Cow()
  .spots(false)
  .color("orange")
.end();
```

```
(cow
   spots #t
   color 'orange)
```

```
cow do
  spots false
  color :orange
end
```

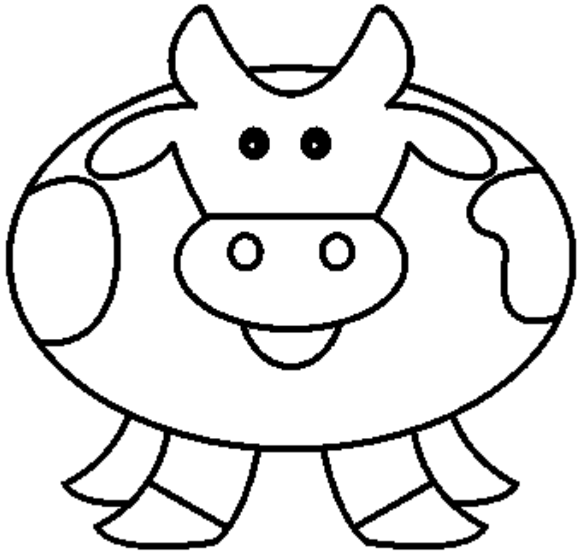
Advantages

- No need to write/maintain parser
- Host language available if needed
- Use of existing tools (IDE) etc.

Drawbacks

- Restricted to host language
- Less static checking
- Fewer opportunities for optimization

External DSLs



```
Cow    ::= “cow” Prop* “end”
Prop   ::= “horns” Bool
         | “spots” Bool
         | “color” Color
Bool   ::= “true” | “false”
Color  ::= “black”
         | “brown”
         | “orange”
```

Repetitio

literal

alternative

Parser generation

yacc
bison
lemon

Grammar

```
Cow   ::= “cow” Prop* “end”
Prop  ::= “horns” Bool
        | “spots” Bool
        | “color” Color
Bool  ::= “true” | “false”
Color ::= “black”
        | “brown”
        | “orange”
```

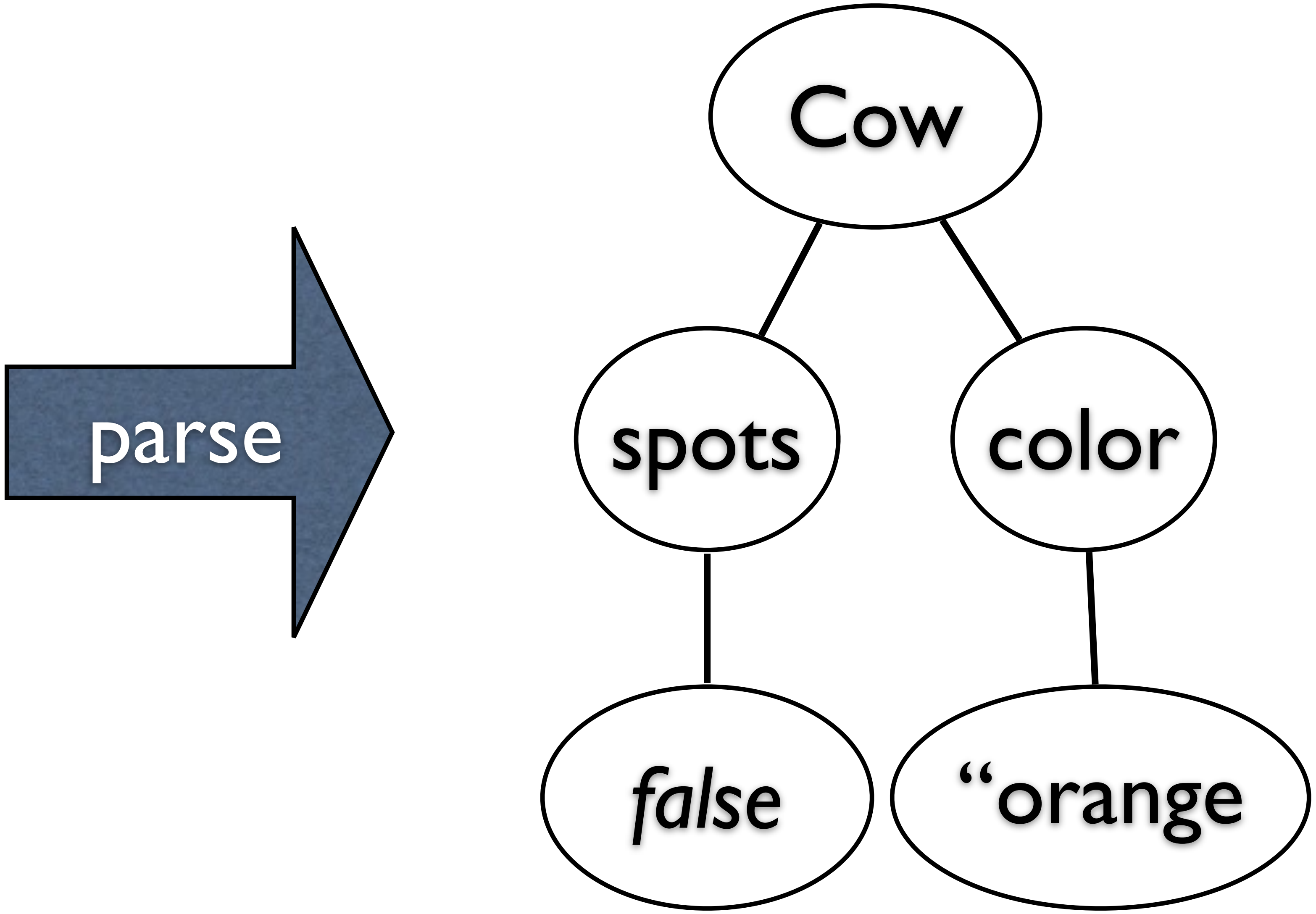
Parser

`parse.exe`

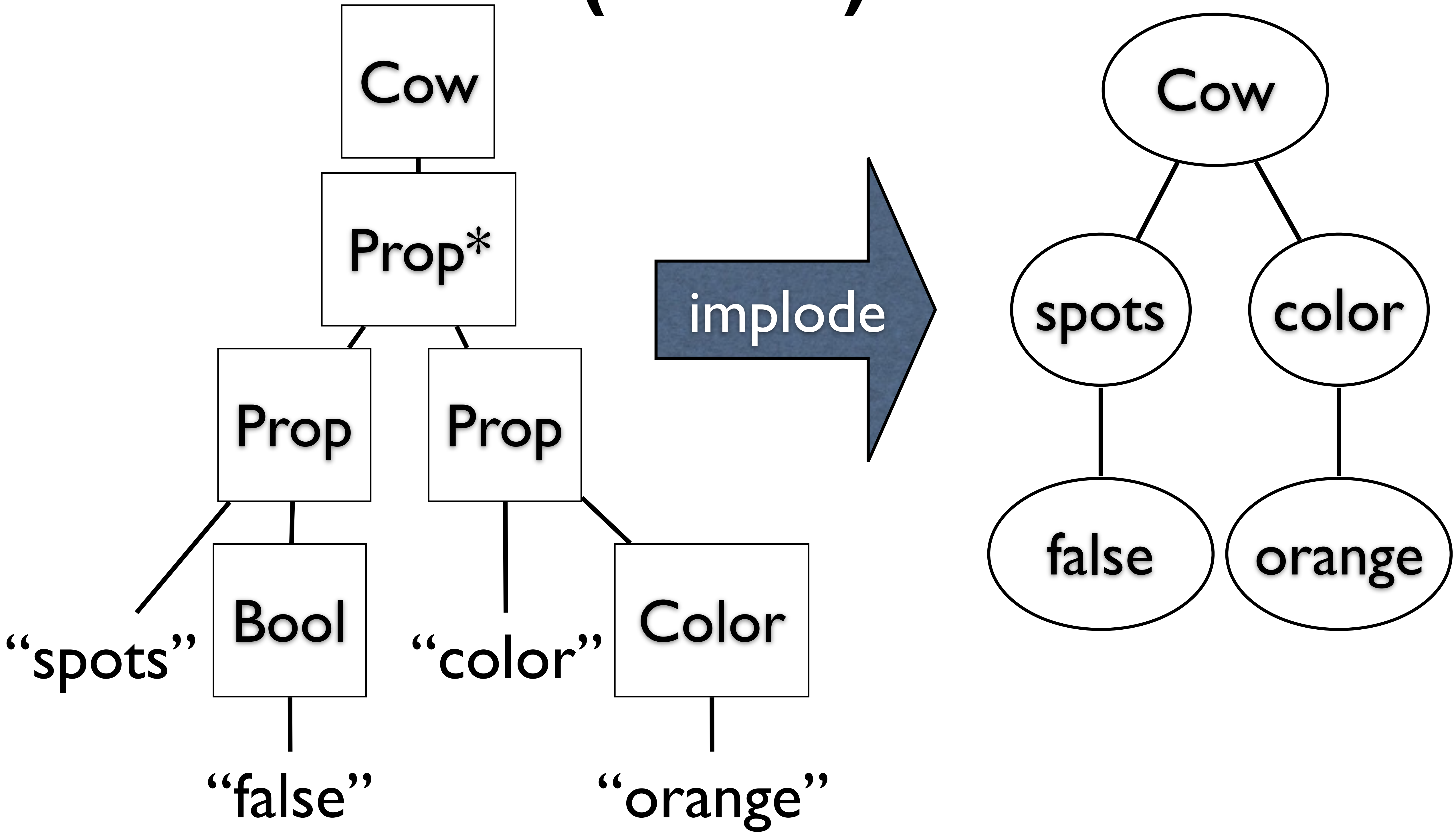
javacup
antlr
Rats!

Parsing

```
cow
  spots false
  color orange
end
```

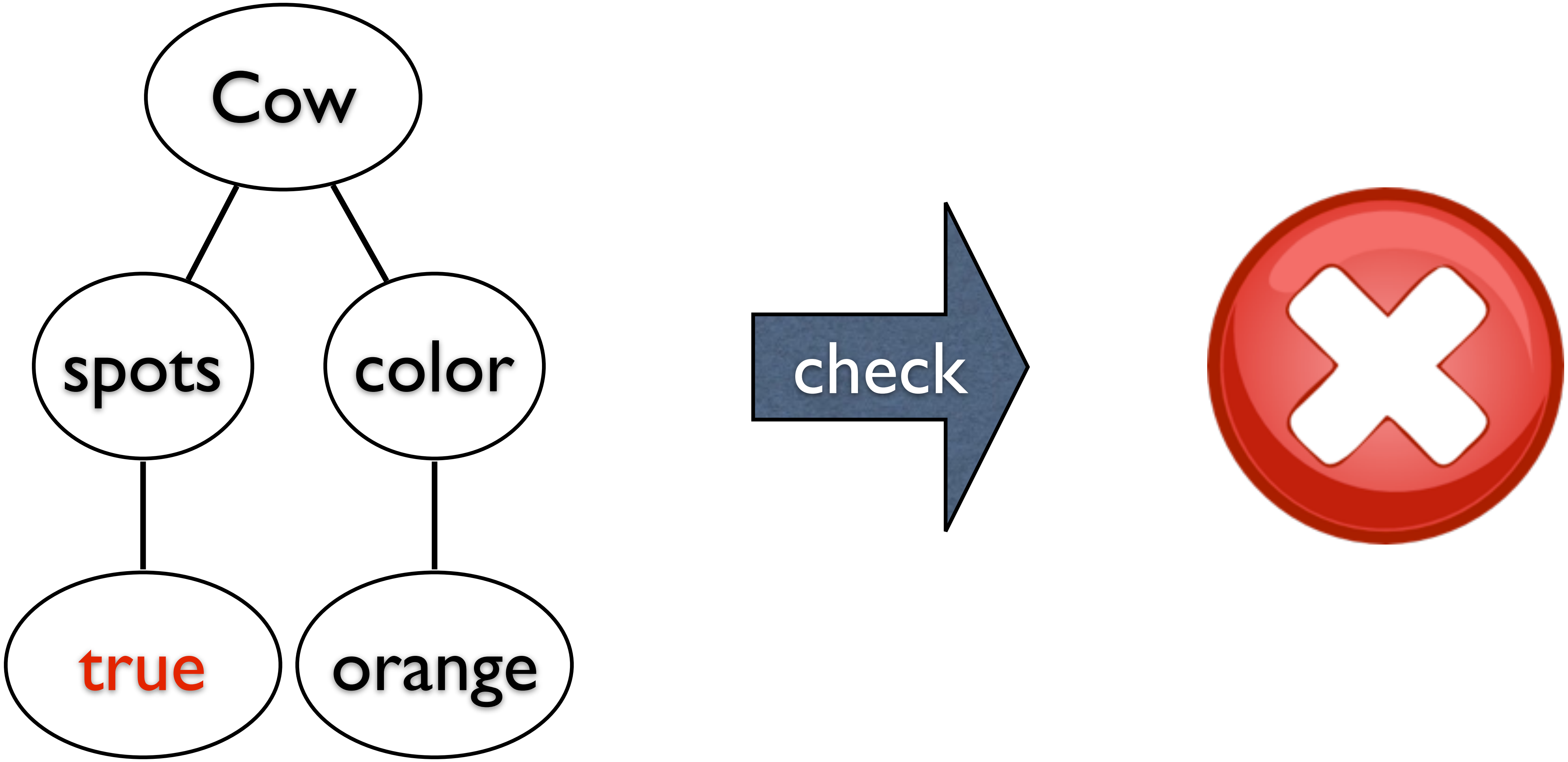


Abstract syntax tree (AST)

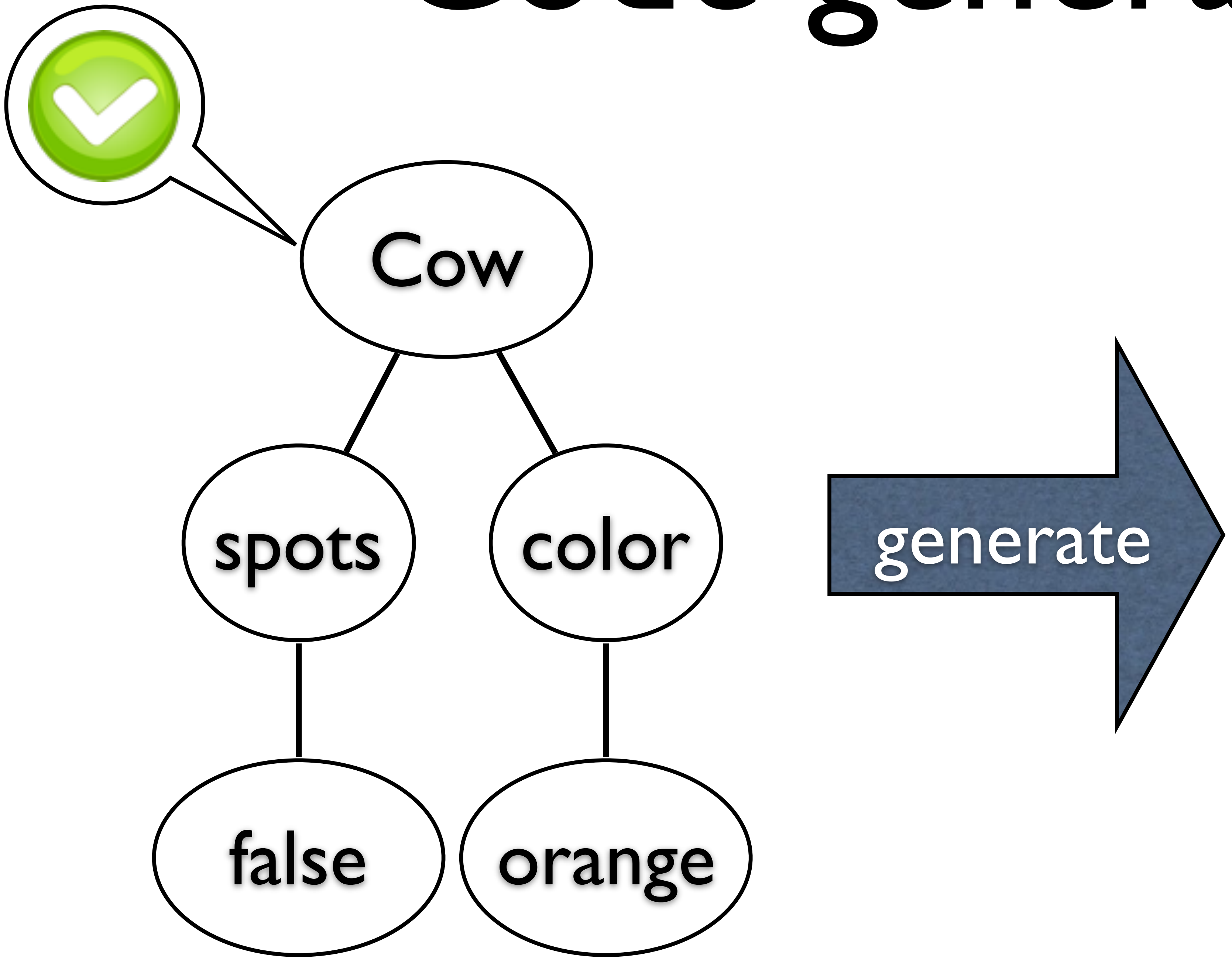


Semantic analysis

Constraint: $spots = \text{true} \Rightarrow color \neq \text{orange}$



Code generation



Cow

spots

color

false

orange

generate

Tools

ANTLR

- Parser generators
- Attribute grammar systems
- Transformation systems
- Language workbenches

MPS

Rascal

Ensō

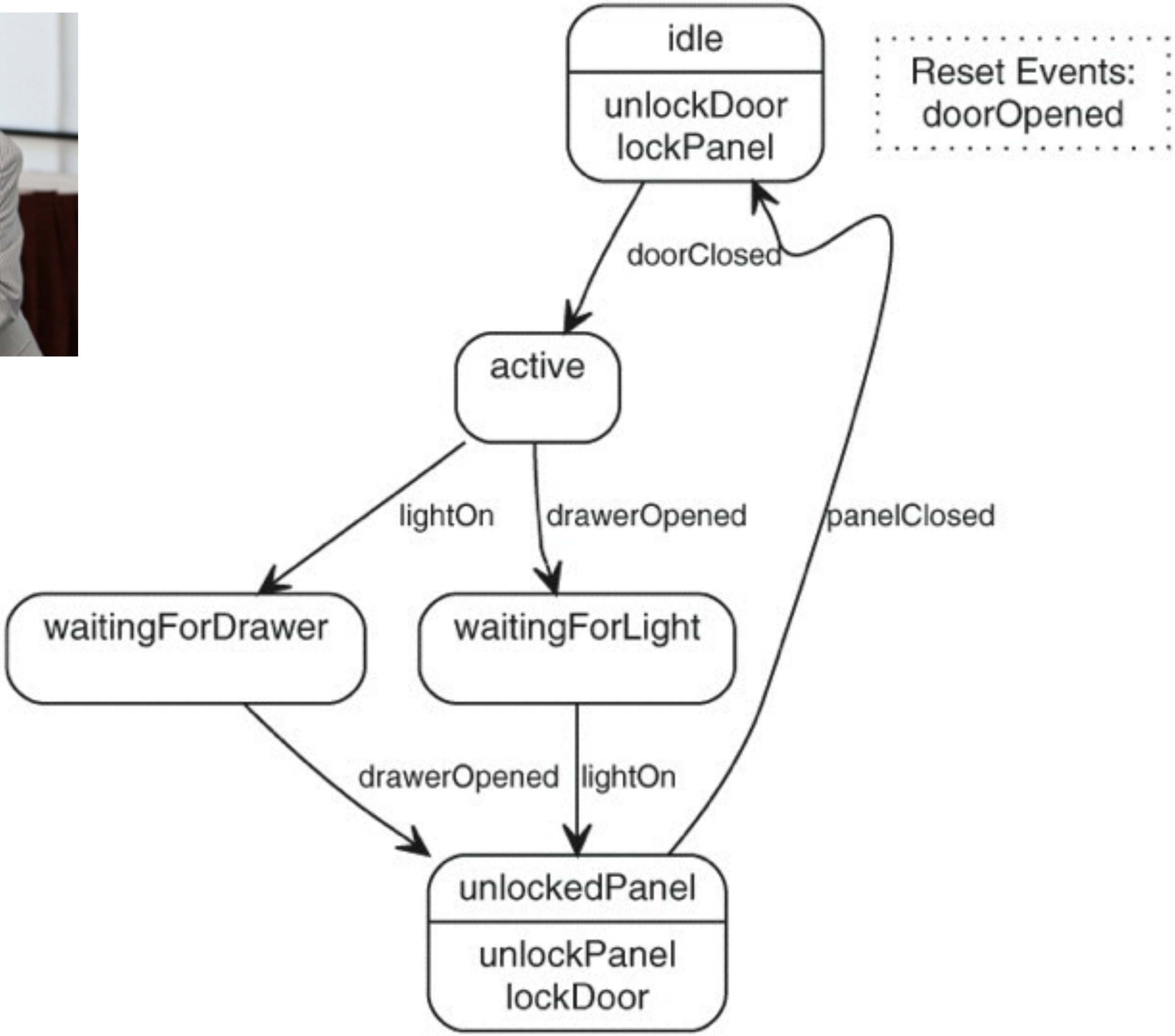
Xtext

Stratego/XT

M "OSLO"

Demo

State machines



idle

unlockDoor
lockPanel

Reset Events:
doorOpened

doorClosed

active

lightOn

drawerOpened

panelClosed

waitingForDrawer

waitingForLight

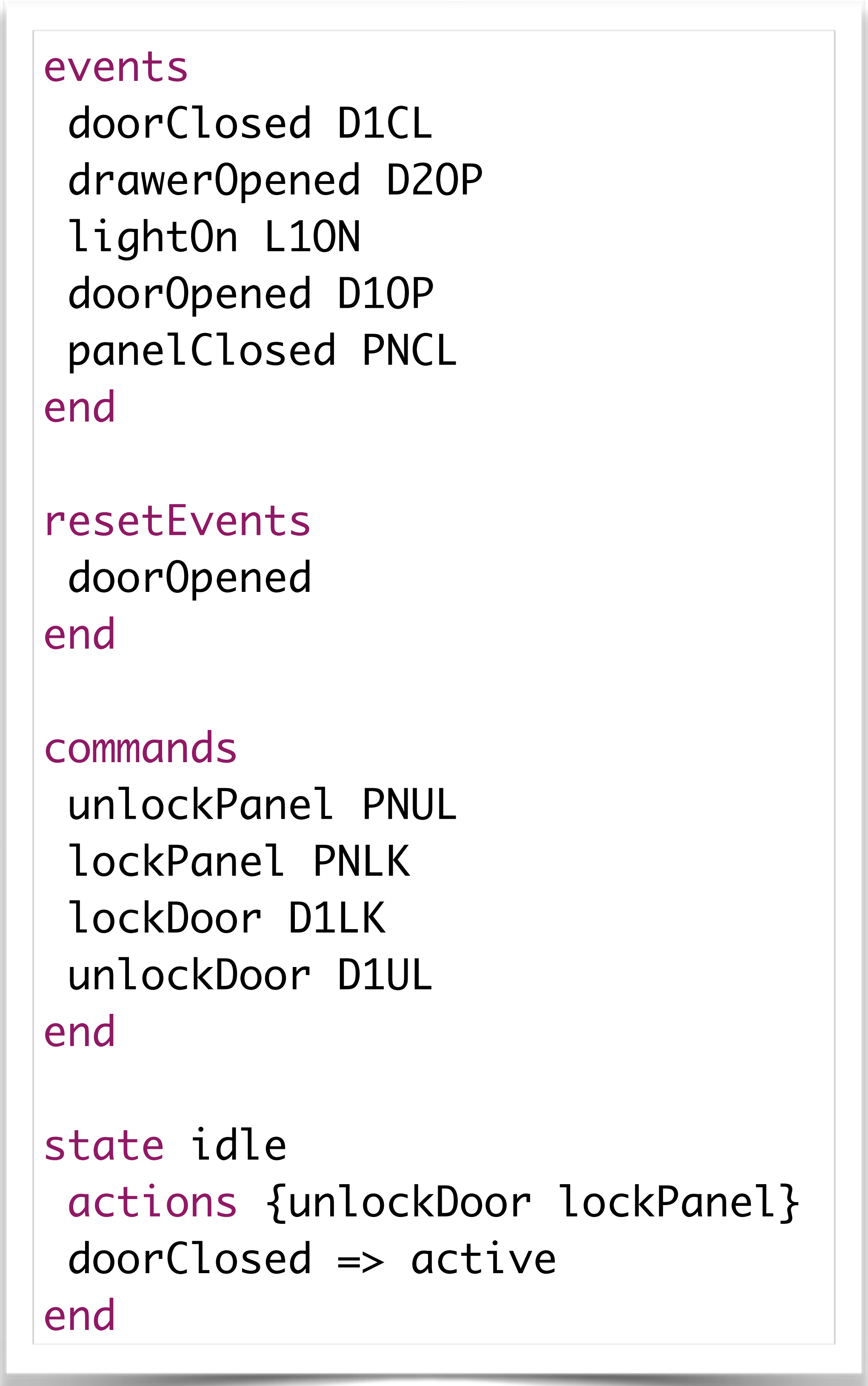
drawerOpened

lightOn

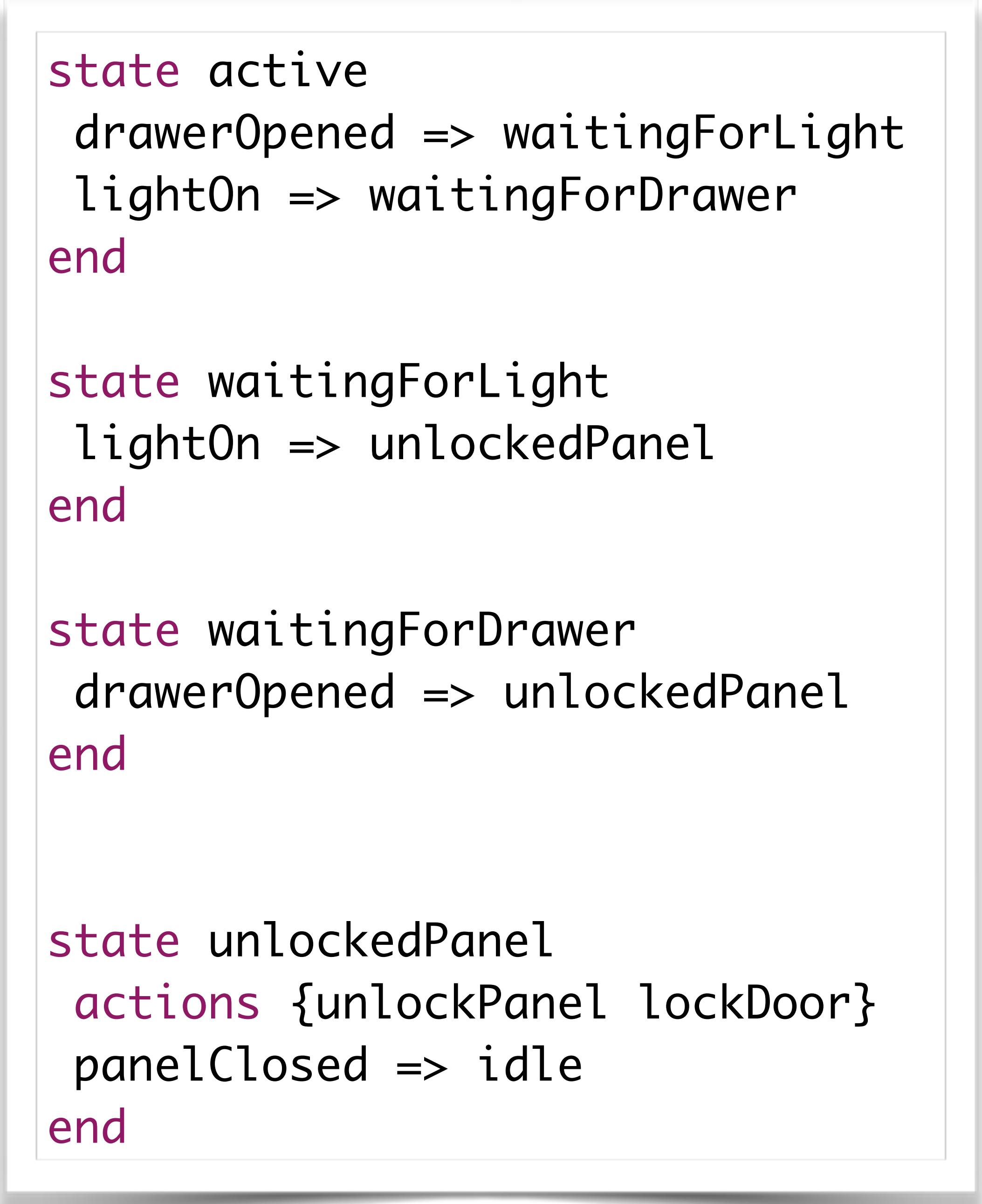
unlockedPanel

unlockPanel
lockDoor

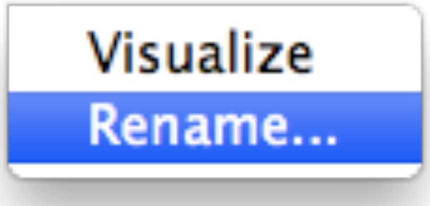
Textual notation

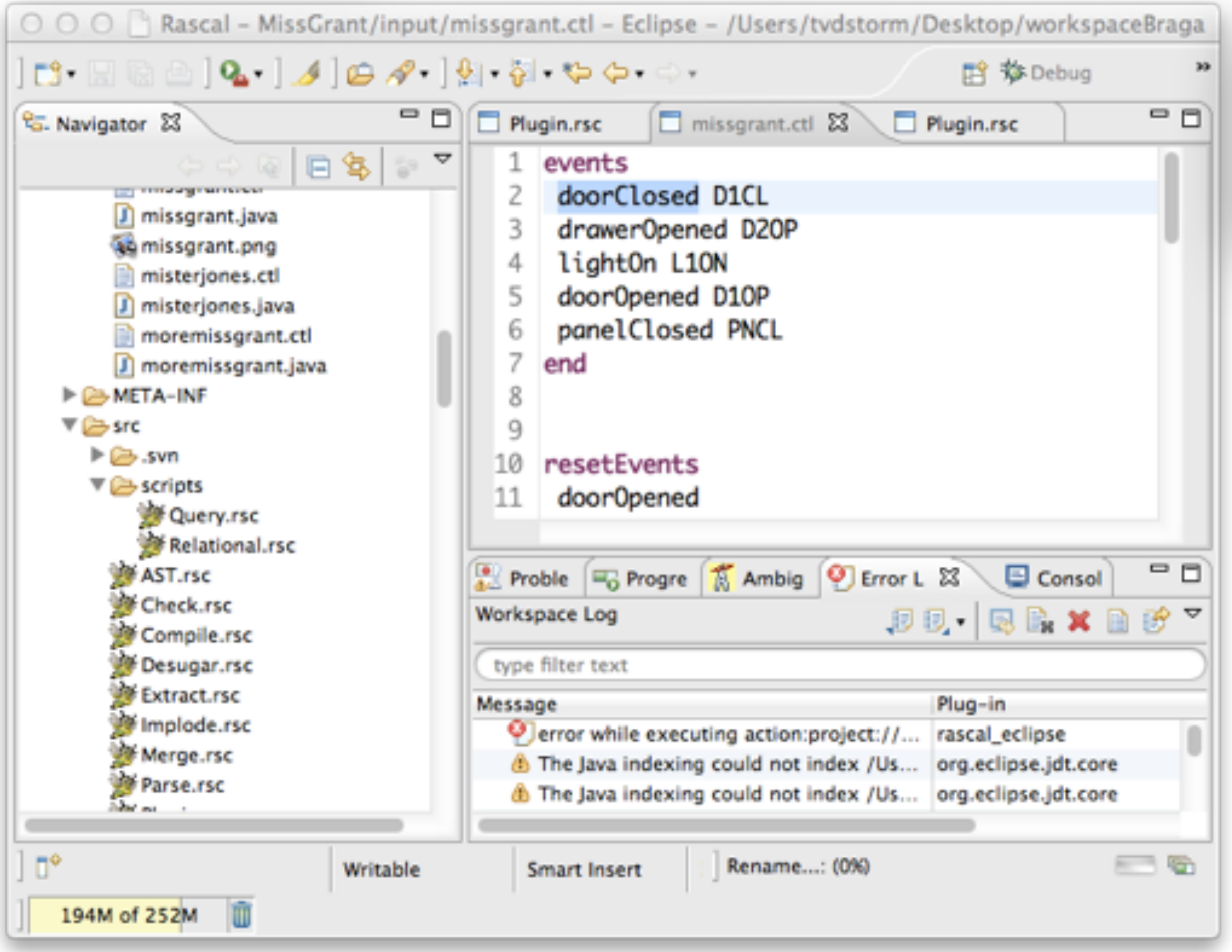


```
events
 doorClosed D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end

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
```

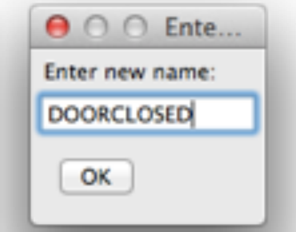




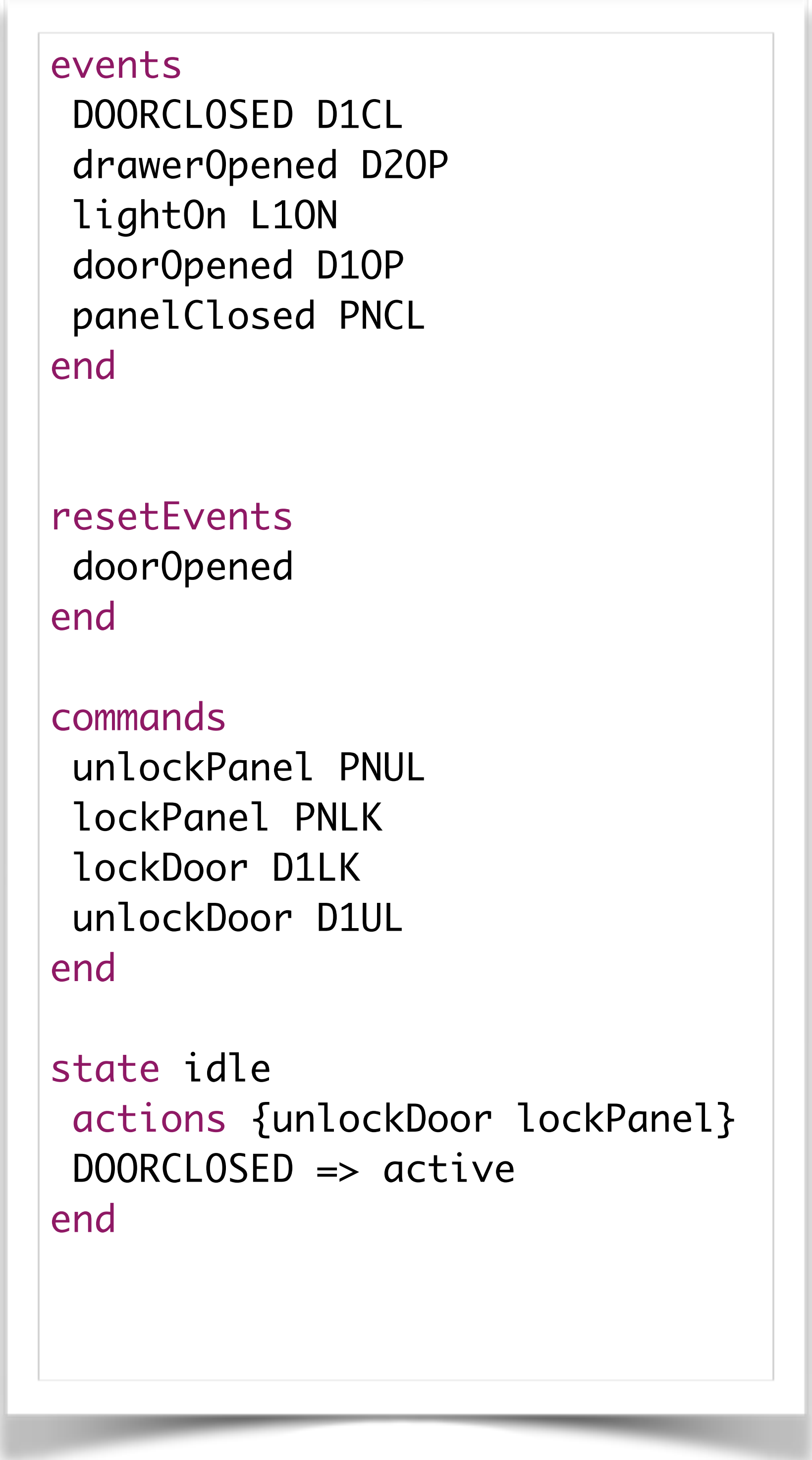
missgrant.java
missgrant.png
misterjones.ctl
misterjones.java
moremissgrant.ctl
moremissgrant.java
META-INF
src
.svn
scripts
Query.rsc
Relational.rsc
AST.rsc
Check.rsc
Compile.rsc
Desugar.rsc
Extract.rsc
Implode.rsc
Merge.rsc
Parse.rsc

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


resetEvents
 doorOpened
```

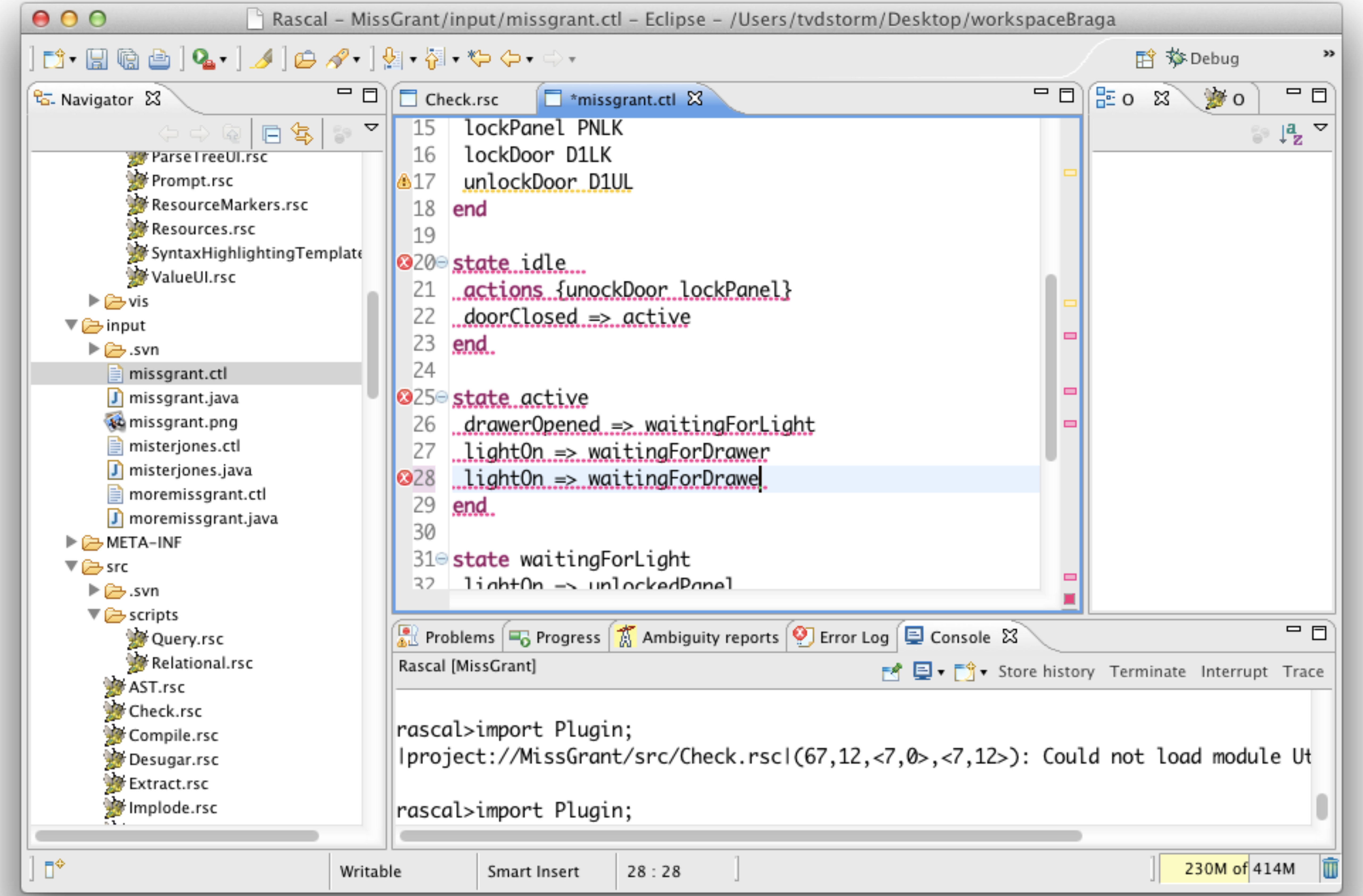
Message	Plug-in
error while executing action:project://...	rascal_eclipse
The Java indexing could not index /Us...	org.eclipse.jdt.core
The Java indexing could not index /Us...	org.eclipse.jdt.core



Enter new name: DOORCLOSED



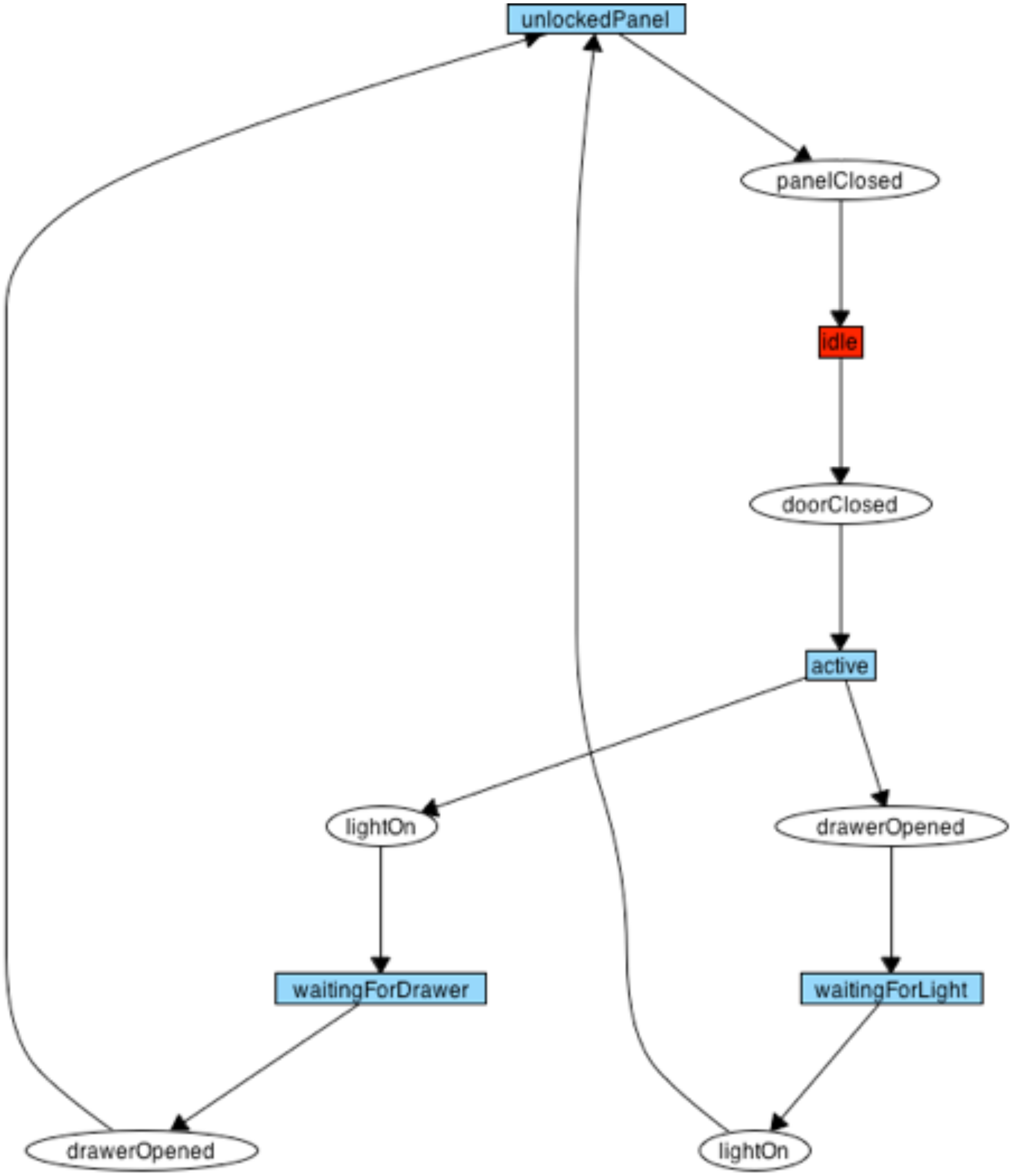
```
events
 DOORCLOSED D1CL
 drawerOpened D2OP
 lightOn L1ON
 doorOpened D1OP
 panelClosed PNCL
end


resetEvents
 doorOpened
end

commands
 unlockPanel PNUL
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

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



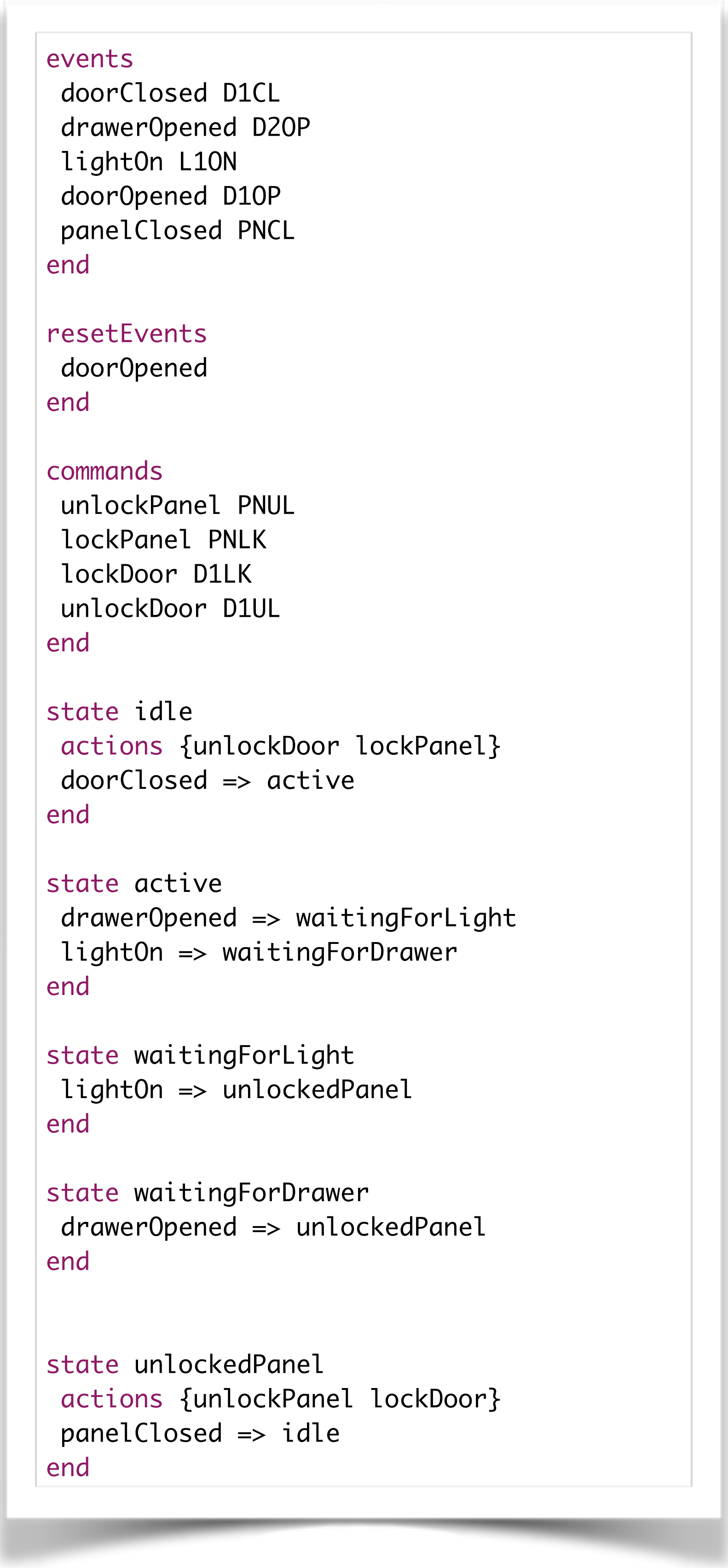
```
 lockPanel PNLK
 lockDoor D1LK
 unlockDoor D1UL
end

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

state active
 drawerOpened => waitingForLight
 lightOn => waitingForDrawer
 lightOn => waitingForDrawe
end

state waitingForLight
 lightOn => unlockedPanel
```

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

rascal>import Plugin;
```

Visualization



Code generation



```
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
```

```
public class missgrant {
    public static void main(String args[]) throws java.io.IOException {
        new missgrant().run(new java.util.Scanner(System.in),
                new java.io.PrintWriter(System.out));
    }

    private static final int state$idle = 0;
    private static final int state$active = 1;
    private static final int state$waitingForLight = 2;
    private static final int state$waitingForDrawer = 3;
    private static final int state$unlockedPanel = 4;

    public void run(java.util.Scanner input, java.io.Writer output)
            throws java.io.IOException {
        int state = state$idle;
        while (true) {
            String token = input.nextLine();
            switch (state) {

            case state$idle: {
                unlockDoor(output);
                lockPanel(output);
                if (doorClosed(token)) {
                    state = state$active;
                }
                if (doorOpened(token)) {
                    state = state$idle;
                }
                break;
            }

            case state$active: {
                if (drawerOpened(token)) {
                    state = state$waitingForLight;
                }
                if (lightOn(token)) {
                    state = state$waitingForDrawer;
                }
                if (doorOpened(token)) {
                    state = state$idle;
                }
                break;
            }

            case state$waitingForLight: {
```

Domain-specific languages

- Better languages, for specific domains
- Capture families of systems
- Higher level of abstraction
- Focus on “what” vs “how”
- Reuse designs, not just code
- Language workbenches (e.g., Rascal)

Take home

- How much code I write is actually relevant for the problem I’m solving?
- What are recurring patterns in the code I’m writing?
- How would I *want* to describe the solution?
- Could I formalize the relevant bits, ... in a DSL?

Thank you

- http://www.rascal-mpl.org
- http://www.cwi.nl/~storm
- storm@cwi.nl
- @tvdstorm