Chapter 13: Visualization Techniques

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning Overview

Licence

Outline

What we want to introduce

- Why visualize?
- How to visualize constraint programs
- Visualization Interface
- Visualization Tool

Background

- Gift grant from Cisco Systems/Silicon Valley Community Foundation
- Cisco owns open-sourced ECLiPSe system
- How to expand user-base?
- Self-taught course in constraint programming
- Intended for Cisco engineers/programmers
- Open source/available to community
- Website
  `http://4c.ucc.ie/~hsimonis/ELearning/index.htm`

Format

- Video lectures
- Slides
- Handout
- Exercises

Problems Handled in Course

- Must have puzzles!
- Send+More=Money
- Sudoku
- N-queens
- Shikaku

Practical Example Problems

- Test plan generation (BIBD)
- Progressive party problem
- Routing and wavelength assignment
- Optical network design
- Car sequencing
- Costas arrays
- Sports scheduling
- Still to come
  - Production scheduling
  - Nurse rostering
  - Airport stand allocation

Intention

- Realistic, life like problems
- Must address scalability issues
- Often, problem not completely specified
- Issue: Hard to verify by hand
- Complexity still limited, not real problems
- No attempt at integration

How do we understand behavior?

- Mental model
- Formal analysis
- Debugging
- Tracing
- Life visualization
- Post-mortem analysis

Why Visualize?

- Understand what is done
- Understand what is done in which order
- Understand what is *not* done
- Understand when to give up

Design Choices

- No deep integration with solver
- Post-mortem visualization
- Intermediate file format
- No view of detailed propagation
  - Tool not intended for debugging constraint engine

Conceptual Model

- Stable state at defined program points
- Granularity
  - Assign value
  - Post constraint
- Show stable state after propagation
- Do not show individual propagation steps

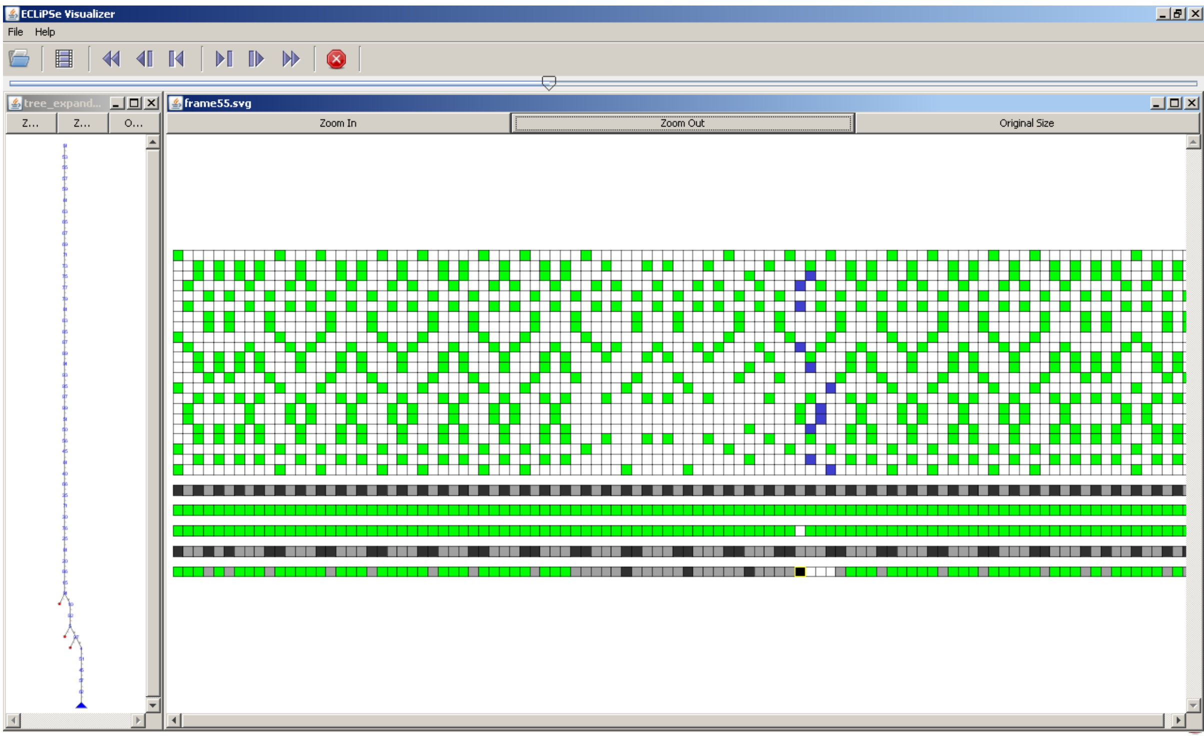
Visualizers

- Search tree
- Variables
- Constraints

Visualization Tool

- Developed in Java
- Show two panes: tree and state
- Navigate along timeline

Visualization Tool: Car Sequencing



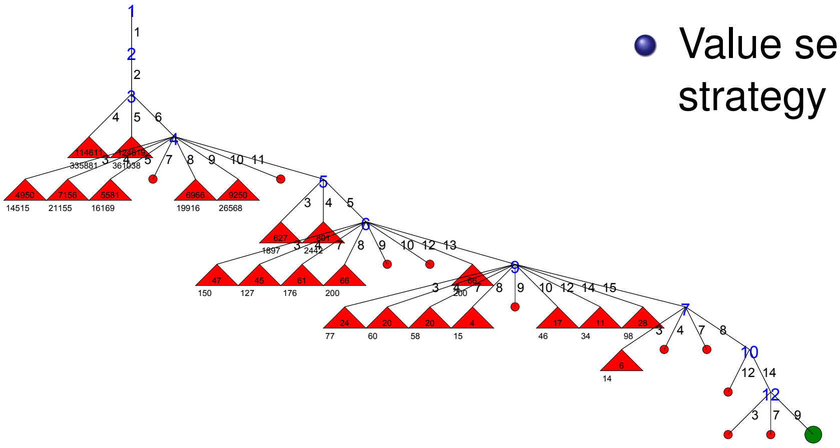
How many visualizers do we need?

- Develop few primitives
  - Cell based view
  - Domain vector
- Allow aggregation
  - Vector/matrix
  - General layout
- Which global constraints require more?
  - Task based view for `cumulative`
  - Matching/flow based representation does not scale

How to Interpret Visualization

- Search tree
  - Good/bad choices
  - Place of backtracking
- State
  - Missing propagation

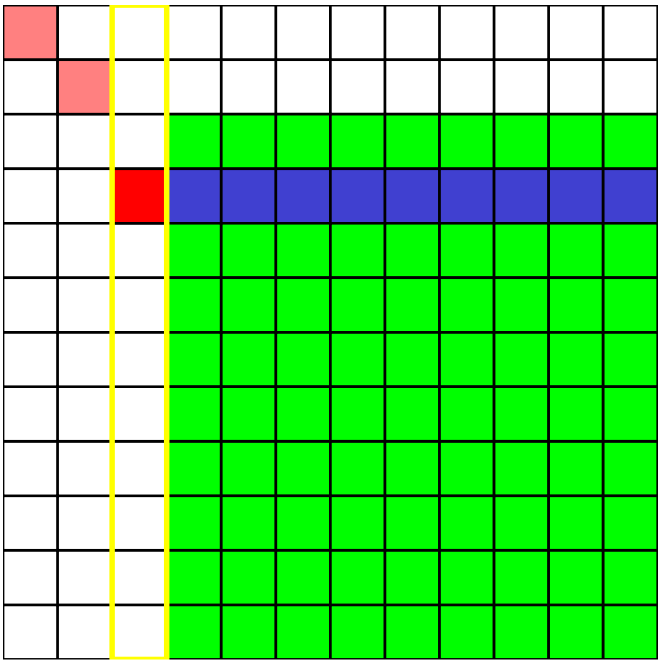
Costas Array Search tree (Size 16)



- Deep backtracking
- Third choice wrong
- Last choice wrong
- Value selection strategy useless

Cork Constraint Computation Centre

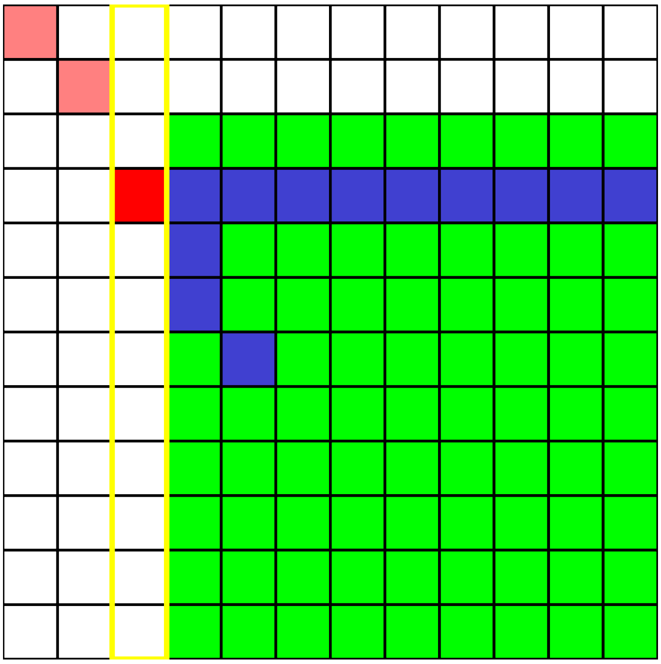
Missing Propagation



The model is doing this

Cork Constraint Computation Centre

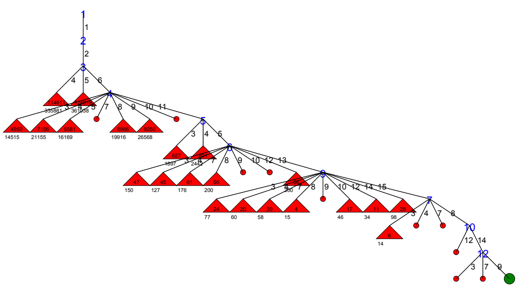
Missing Propagation



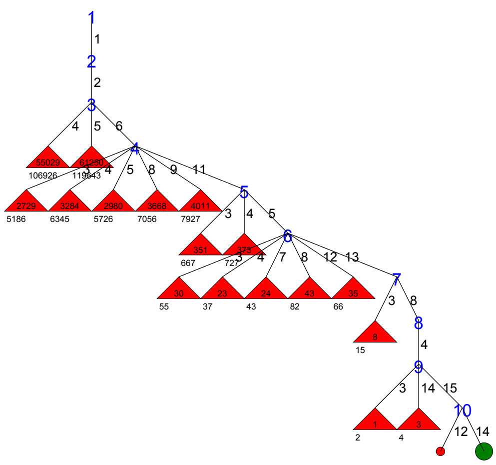
It could be doing that!

Comparison (Search Tree, size 16)

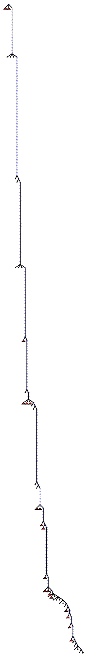
Initial Model



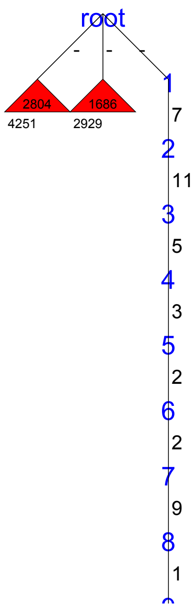
Improved Model



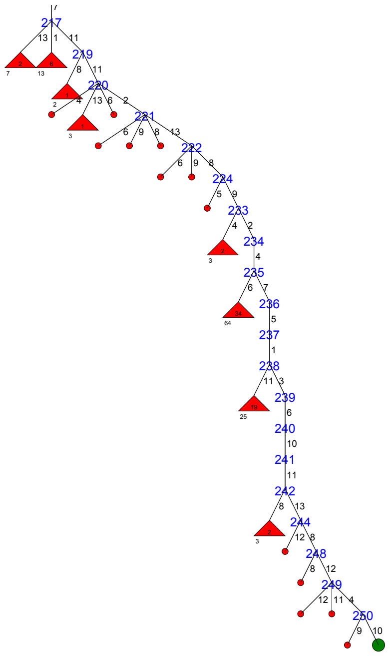
Progressive Party Problem, 9 Time Periods



2 Restarts Before Solution Found



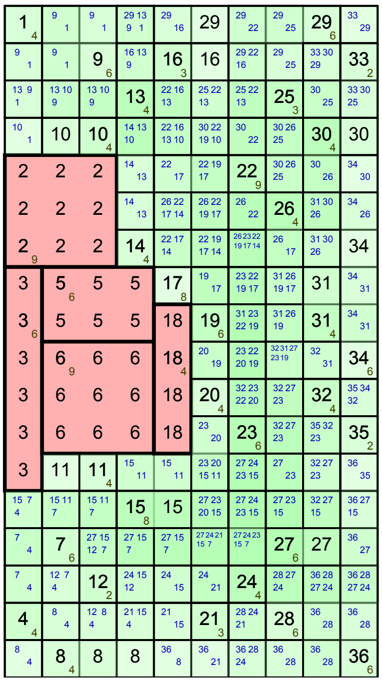
Value Choice Strategy Not Focused



Progressive Party

- Clearly impossible to explore search space
- Either many solutions or good value selection
- Value selection at end rather poor
- Probably many solutions

Missing Propagation: Shikaku



Sendmore Program Annotated

```
sendmory(L,Output,IgnoreFixed):-
  L=[S,E,N,D,M,O,R,Y],
  L :: 0..9,
  create_visualization([output:Output,
                        ignore_fixed:IgnoreFixed,
                        width:8,
                        height:10],Handle),
  add_visualizer(Handle,
                 vector(L),
                 [display:expanded]),
  alldifferent(L),draw_visualization(Handle),
  S #\= 0,draw_visualization(Handle),
  M #\= 0,draw_visualization(Handle),
```

Sendmore Program Annotated

```
1000*S + 100*E + 10*N + D +
1000*M + 100*O + 10*R + E #=
10000*M + 1000*O + 100*N + 10*E + Y,
name_variables(Handle,L,
               ['S','E','N','D','M','O','R','Y'],
               Pairs),
root(Handle),
search(Pairs,1,input_order,
       tree_indomain(Handle,_),
       complete,[]),
solution(Handle),
close_visualization(Handle).
```

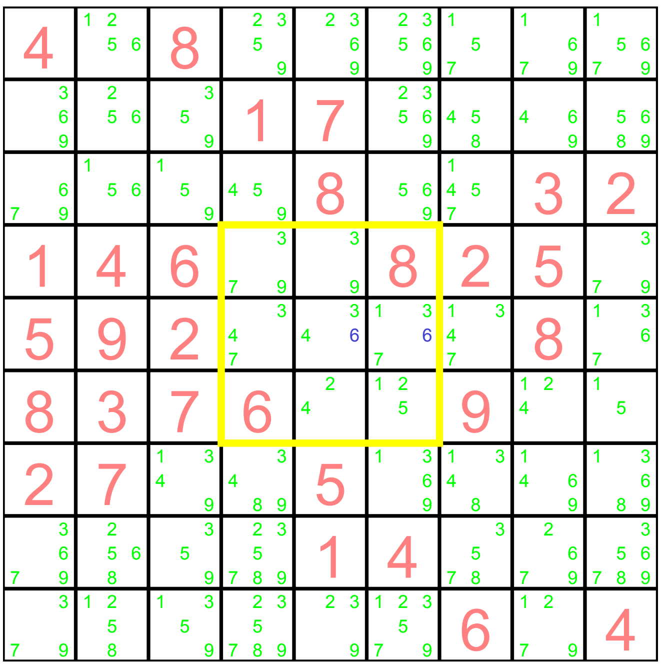

Sudoku Program Annotated

```
model(Matrix,Method,Output):-
  Matrix[1..9,1..9] :: 1..9,
  create_visualization([output:Output,
                        width:9,
                        height:9],Handle),
  add_visualizer(Handle,
                 domain_matrix(Matrix),
                 [display:text]),
  draw_visualization(Handle),
  (for(I,1,9),
   param(Matrix,Method,Handle) do
      Method:alldifferent(Matrix[I,1..9]),
      draw_visualization(Handle,[focus:row(I)]),
      Method:alldifferent(Matrix[1..9,I]),
      draw_visualization(Handle,[focus:col(I)])
  ),
```

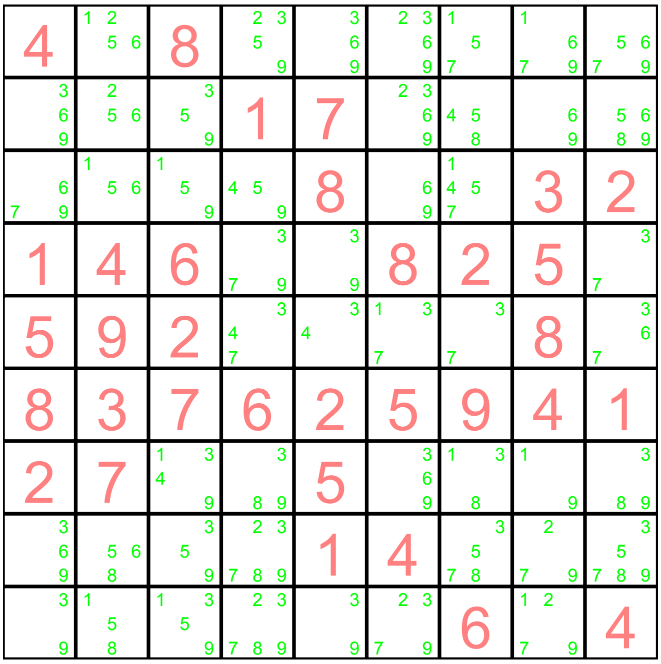
Sudoku Program Annotated

```
(multifor([I,J],[1,1],[7,7],[3,3]),
 param(Matrix,Method,Handle) do
    Method:alldifferent(Matrix[I..I+2,J..J+2]),
    draw_visualization(Handle,
                        [focus:block(I,J,3,3)])
),
extract_array(Handle,row,Matrix,NamedList),
root(Handle),
search(NamedList,1,input_order,
        tree_indomain(Handle,_),
        complete,[]),
solution(Handle),
close_visualization(Handle).
```

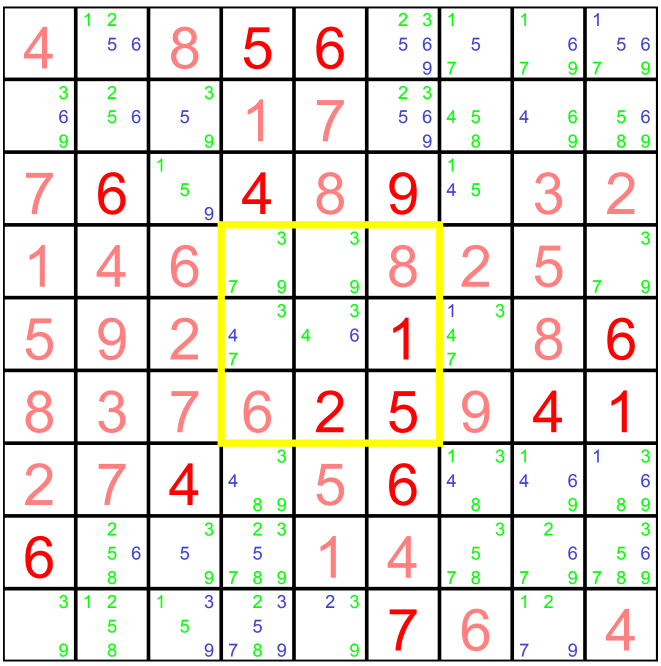

Propagation Steps (Forward Checking)



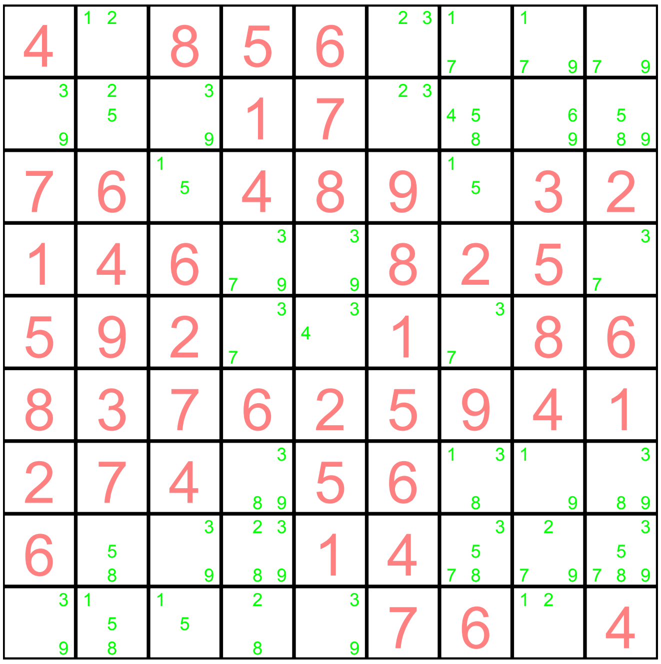
After Setup (Forward Checking)



Propagation Steps (Bounds Consistency)



After Setup (Bounds Consistency)

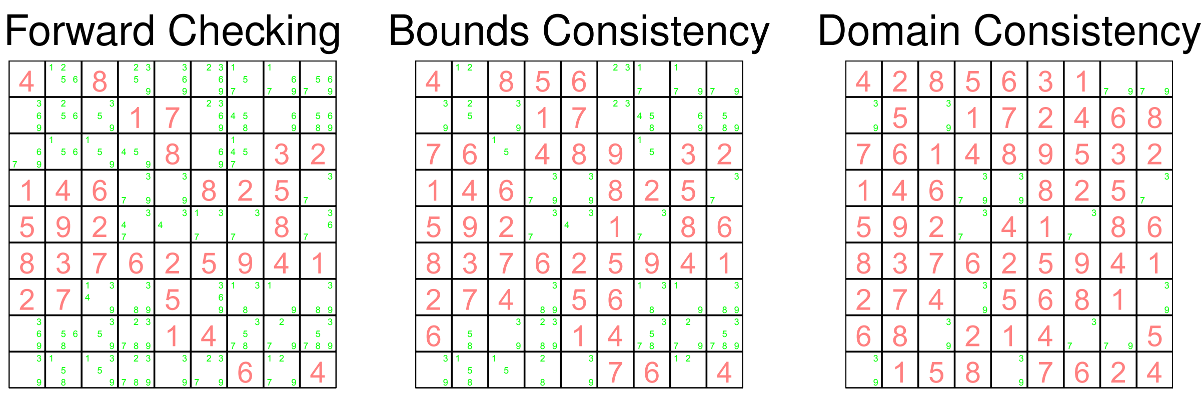


Propagation Steps (Domain Consistency)

After Setup (Domain Consistency)

4	2	8	5	6	3	1	7 9	7 9
3 9	5	3 9	1	7	2	4	6	8
7	6	1	4	8	9	5	3	2
1	4	6	3 7 9	3 9	8	2	5	3 7
5	9	2	3 7	4	1	3 7	8	6
8	3	7	6	2	5	9	4	1
2	7	4	3 9	5	6	8	1	3 9
6	8	3 9	2	1	4	3 7	7 9	5
3 9	1	5	8	3 9	7	6	2	4

Comparison



Instrumented indomain

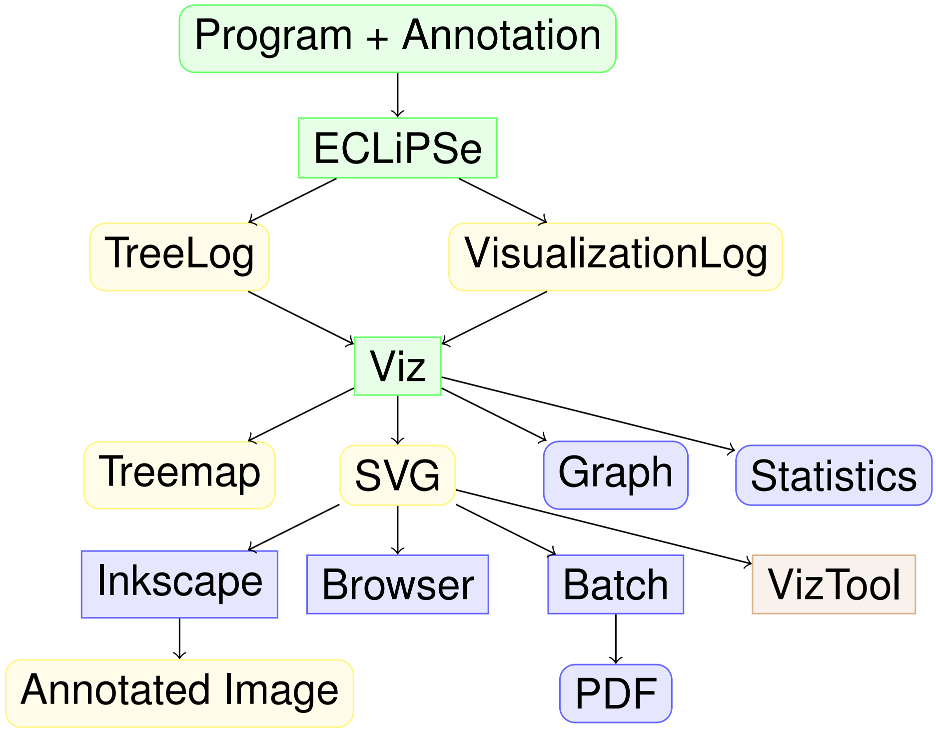
```
tree_indomain_generic(Term,Handle,Handle,Type):-
  Handle = visualization{ignore_fixed:IgnoreFixed,
                          var_arg:VarArg,
                          name_arg:NameArg,
                          focus_arg:FocusArg},
  arg(VarArg,Term,X),
  ((integer(X),IgnoreFixed = yes) ->
      true
  ;
      arg(NameArg,Term,Name),
      arg(FocusArg,Term,Focus),
      get_domain_as_list(X,L),
      get_domain_size(X,Size),
      reorganize_domain(X,L,Type,K),
      try_value(Handle,X,K,Name,Size,Focus)
  ).
```


Introduction
Visualization by Annotation
Visualization Interface
Conclusions

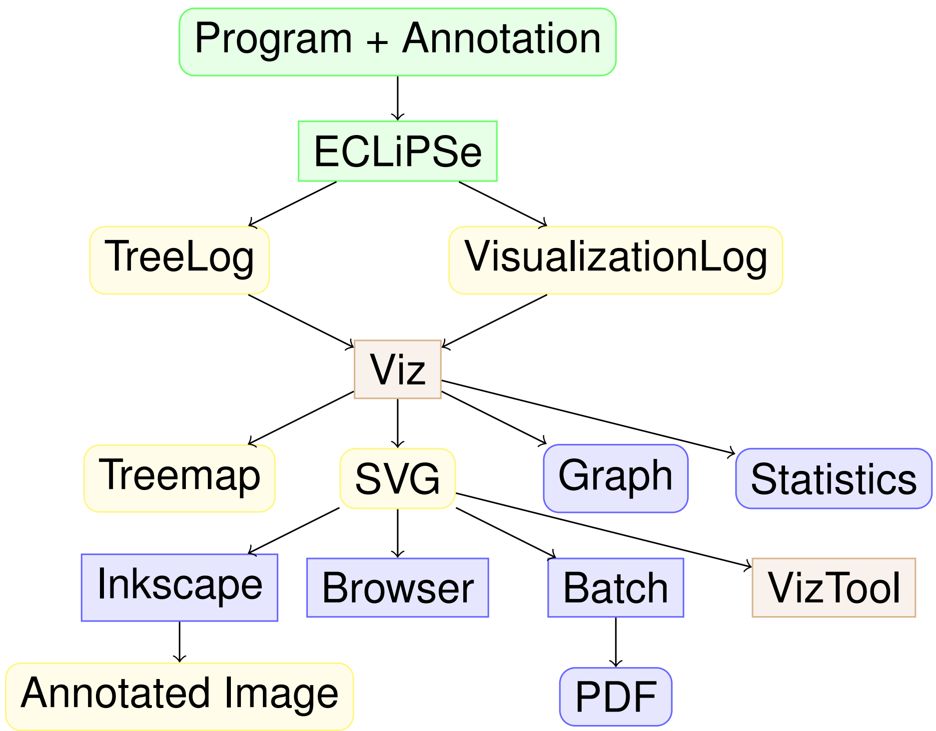
Instrumented indomain

```
try_value(Handle,X,[V|_],Name,Size,Focus):-
  ((X = V, true) ->
      try(Handle,Name,Size,V),
      focus_option(Focus,FocusOption),
      draw_visualization(Handle,FocusOption)
  ;
      failure(Handle,Name,Size,V),
      fail_option(Focus,V,FailOption),
      draw_visualization(Handle,FailOption),
      fail
  ).
try_value(Handle,X,[_|R],Name,Size,Focus):-
  try_value(Handle,X,R,Name,Size,Focus).
```

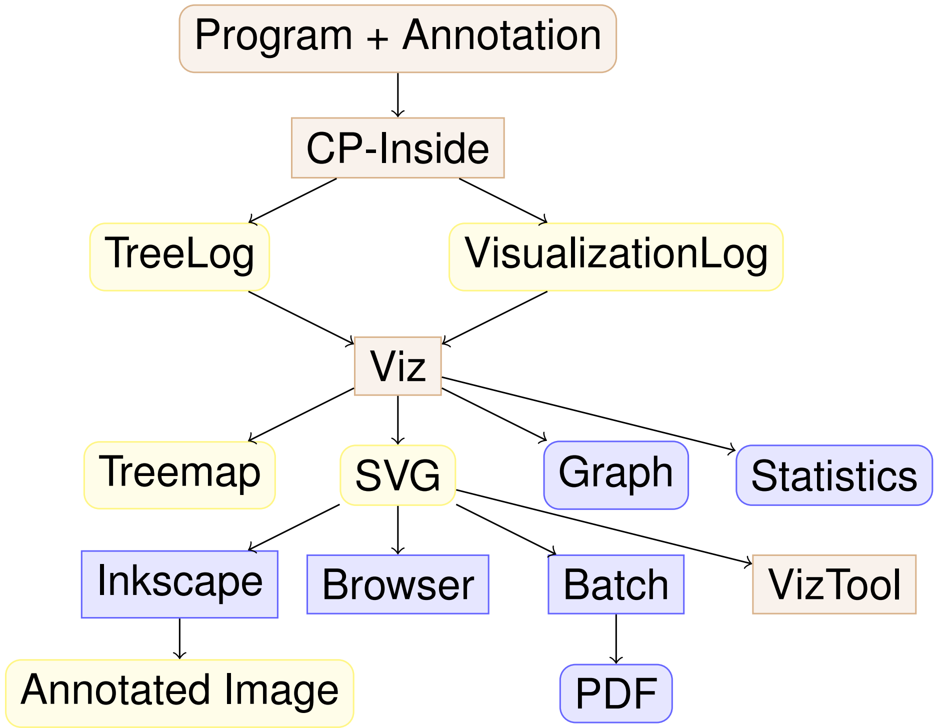
Architecture (Current)



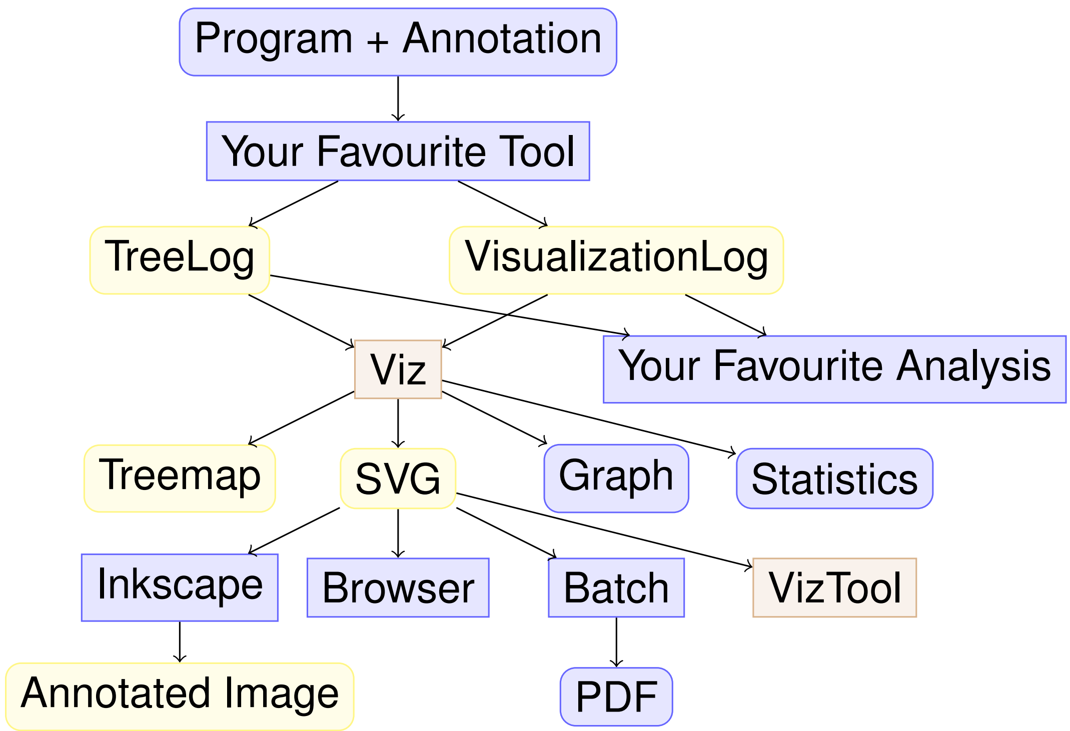
Architecture (Planned)



CP-Inside



Generic Tool

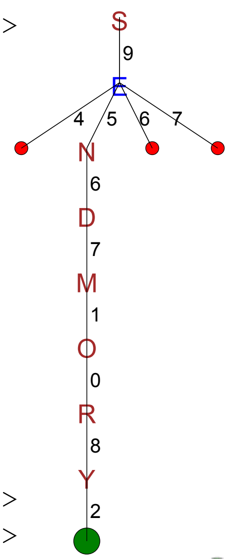


TreeLog Format

- XML based description
- Record information about nodes in search tree
  - Choices
  - Failures
  - Success
- Redundant information to ease generation

TreeLog Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tree version="1.0" >
<root id="0"/>
<try id="1" parent="0" name="S" size="1" value="9" />
<fail id="2" parent="1" name="E" size="4" value="4" />
<try id="3" parent="1" name="E" size="4" value="5" />
<try id="4" parent="3" name="N" size="1" value="6" />
<try id="5" parent="4" name="D" size="1" value="7" />
<try id="6" parent="5" name="M" size="1" value="1" />
<try id="7" parent="6" name="O" size="1" value="0" />
<try id="8" parent="7" name="R" size="1" value="8" />
<try id="9" parent="8" name="Y" size="1" value="2" />
<succ id="9"/>
<fail id="10" parent="1" name="E" size="4" value="6" />
<fail id="11" parent="1" name="E" size="4" value="7" />
</tree>
```



VisualizerLog Format

- XML based description
- Describe state of variables and/or constraints at specific stages
  - Where annotated in program
  - For every node in tree
- Linked to search tree log

VisualizerLog Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<visualization version="1.0" >
<visualizer id="1" type="vector" display="expanded" x="0" y="0"
  width="8" height="10" group="1" min="0"  max="9" />
<state id="1" tree_node="-1" >
<visualizer_state id="1" >
<dvar index="1" domain="0 .. 9" />
<dvar index="2" domain="0 .. 9" />
<dvar index="3" domain="0 .. 9" />
<dvar index="4" domain="0 .. 9" />
<dvar index="5" domain="0 .. 9" />
<dvar index="6" domain="0 .. 9" />
<dvar index="7" domain="0 .. 9" />
<dvar index="8" domain="0 .. 9" />
</visualizer_state>
</state>
...
```

VisualizerLog Example

```
...
<state id="2" tree_node="-1" >
<visualizer_state id="1" >
<dvar index="1" domain="1 .. 9" />
<dvar index="2" domain="0 .. 9" />
<dvar index="3" domain="0 .. 9" />
<dvar index="4" domain="0 .. 9" />
<dvar index="5" domain="0 .. 9" />
<dvar index="6" domain="0 .. 9" />
<dvar index="7" domain="0 .. 9" />
<dvar index="8" domain="0 .. 9" />
</visualizer_state>
</state>
...
```


VisualizerLog Example

```
...
<state id="5" tree_node="1" >
<visualizer_state id="1" >
<integer index="1" value="9" />
<dvar index="2" domain="4 .. 7" />
<dvar index="3" domain="5 .. 8" />
<dvar index="4" domain="2 .. 8" />
<integer index="5" value="1" />
<integer index="6" value="0" />
<dvar index="7" domain="2 .. 8" />
<dvar index="8" domain="2 .. 8" />
<focus group="-" index="1" />
</visualizer_state>
</state>
...
```

VisualizerLog Example

```
...
<state id="6" tree_node="2" >
<visualizer_state id="1" >
<integer index="1" value="9" />
<dvar index="2" domain="4 .. 7" />
<dvar index="3" domain="5 .. 8" />
<dvar index="4" domain="2 .. 8" />
<integer index="5" value="1" />
<integer index="6" value="0" />
<dvar index="7" domain="2 .. 8" />
<dvar index="8" domain="2 .. 8" />
<failed group="-" index="2" value="4" />
</visualizer_state>
</state>
...
```

Cork Constraint Computation Centre

VisualizerLog Example

```
...
<state id="14" tree_node="9" >
<visualizer_state id="1" >
<integer index="1" value="9" />
<integer index="2" value="5" />
<integer index="3" value="6" />
<integer index="4" value="7" />
<integer index="5" value="1" />
<integer index="6" value="0" />
<integer index="7" value="8" />
<integer index="8" value="2" />
</visualizer_state>
</state>
...
</visualization>
```

Cork Constraint Computation Centre

Conclusions

- New ELearning course for ECLiPSe
- Open source material, Creative Commons BY-NC-SA license
  - Application driven
  - Modelling with global constraints
  - Customizing search
- Effort only justifiable through Cisco grant

Visualization

- Design choice: System independent
- Provide enough information to user of system, not to tool developer
- Relatively few primitives, extensible for specific global constraints
- XML intermediate format, open for specific analysis