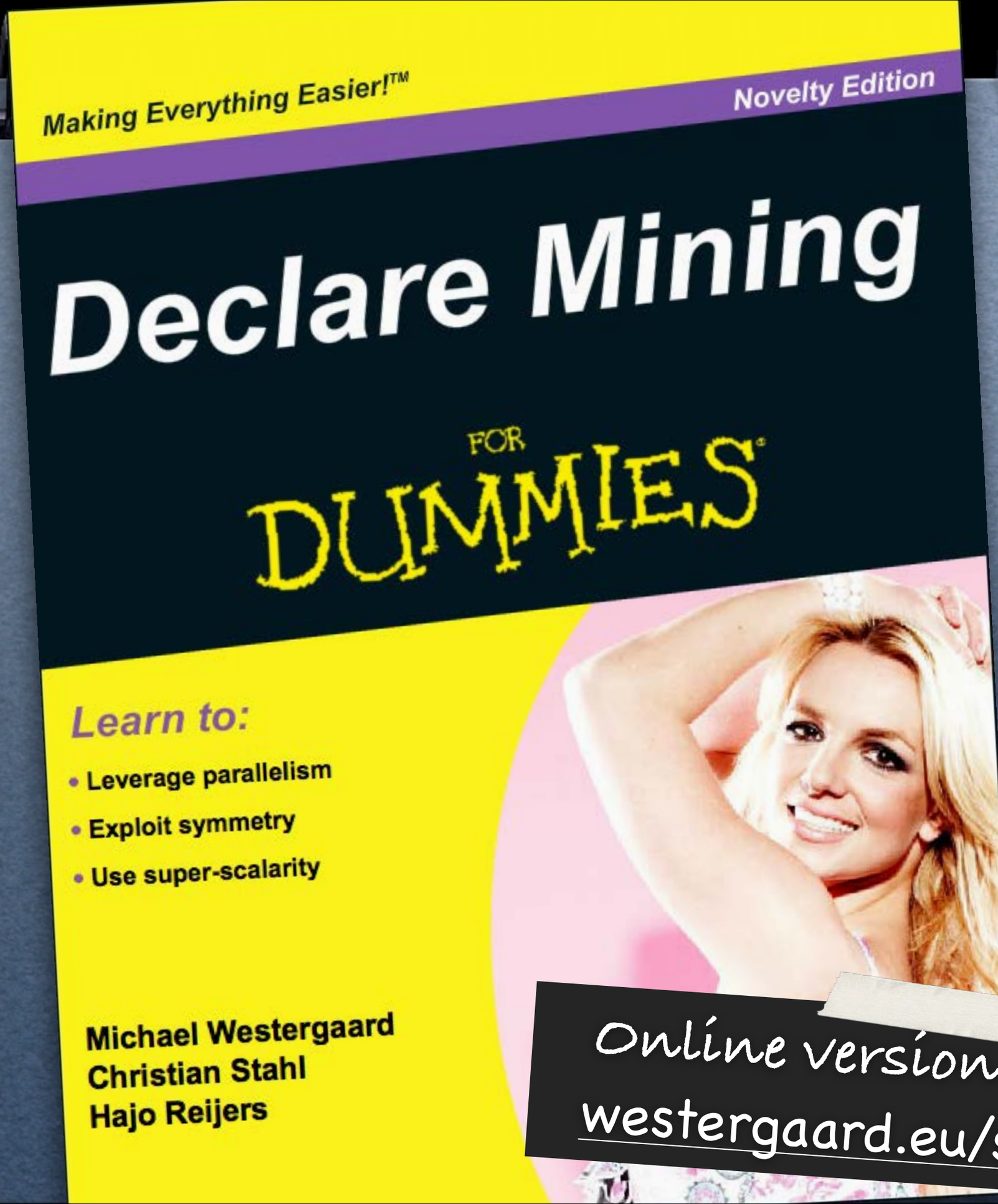
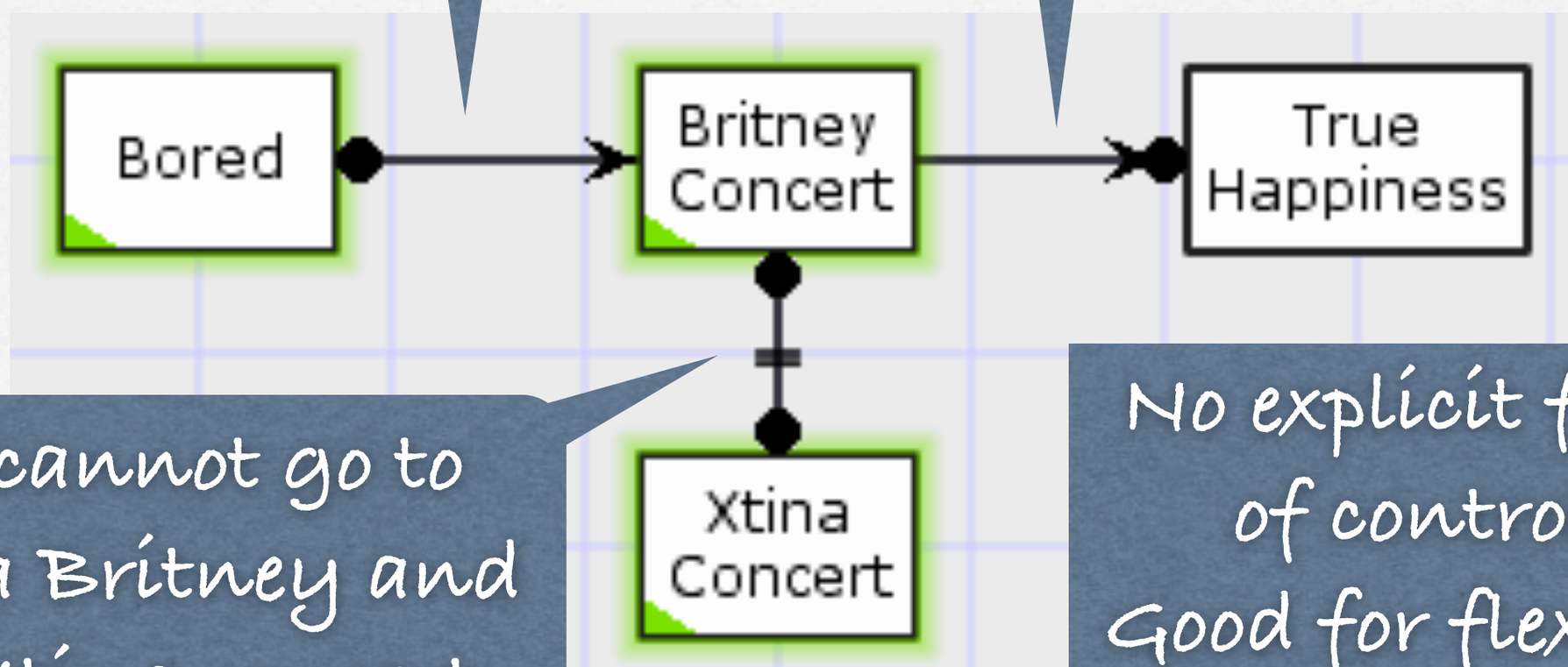


Online version of slides:
westergaard.eu/short12157

Declarative Models

If bored, we have to go to a Britney concert

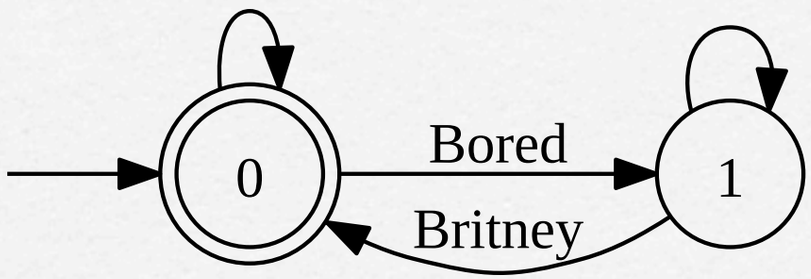
We can only be truly happy after going to a Britney concert

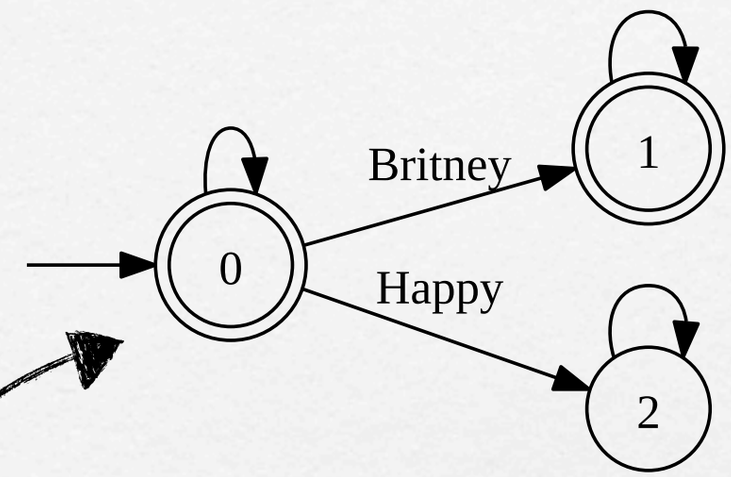


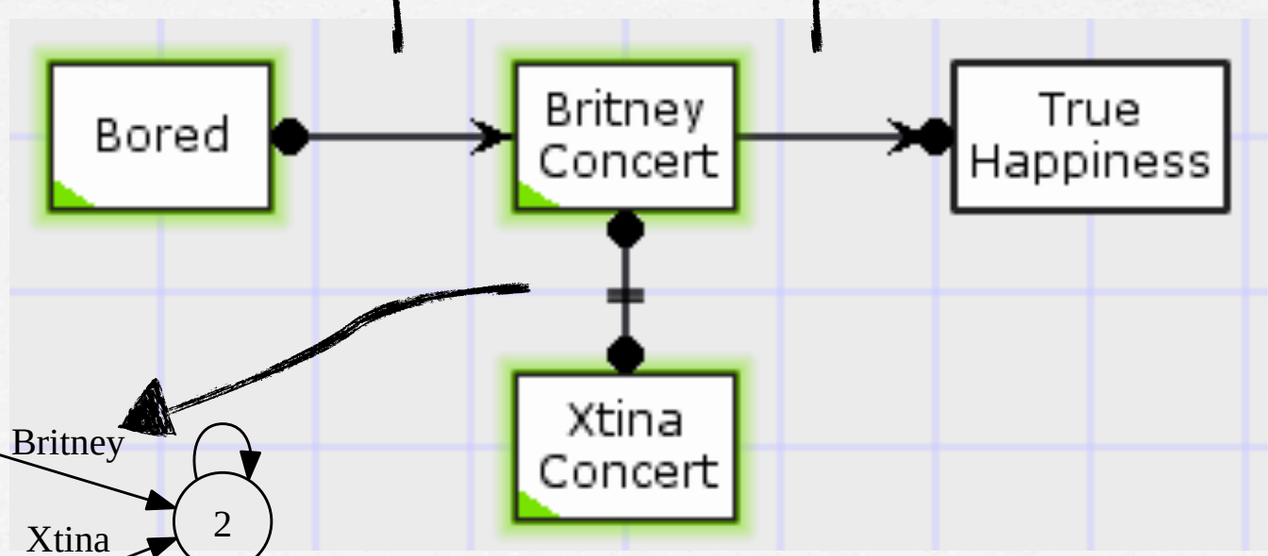
We cannot go to both a Britney and an Xtina concert

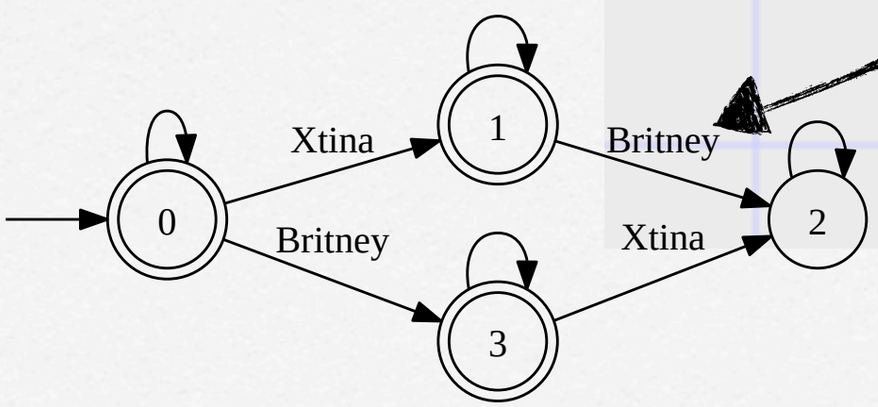
No explicit flow of control
Good for flexible processes

Declarative Mining









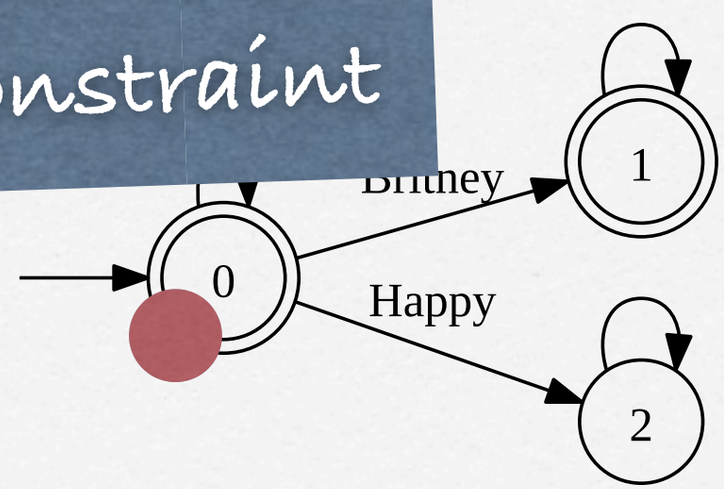
Declarative Mining

Bored, Britney, Happy

Bored, Beer, Happy

REJECTED

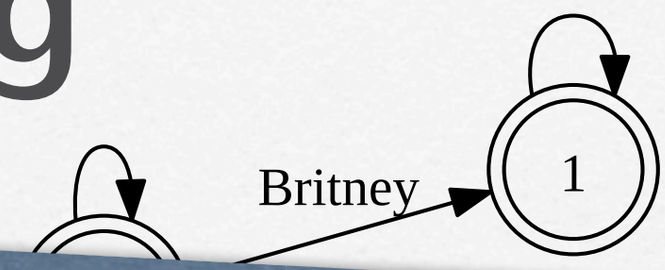
If we accept enough (say ≥ 50%) traces, we accept the constraint



Declarative Mining

Bored, Britney, Happy

Bored, Beer, Happy



Realistic: $n \geq 30$

Sometimes: $p = 5$

Then:

$30 \cdot 29 \cdot 28 \cdot 27 \cdot 26 =$

17.100.720 instantiations

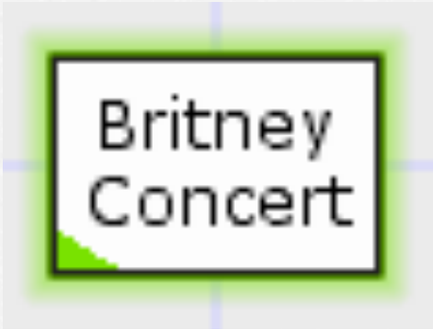
- 2 parameters (p)
- 4 event classes (n)
- $4 \cdot 3 = 12$ instantiations
- In general $n \cdot (n-1) \cdot \ldots \cdot (n - p + 1)$

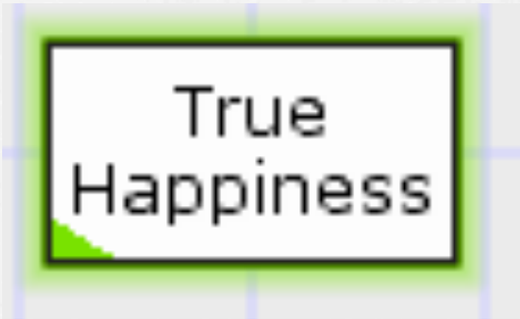
Classic Solution

Contraposition

- If 

often (say ≥ 50%)

- Then 

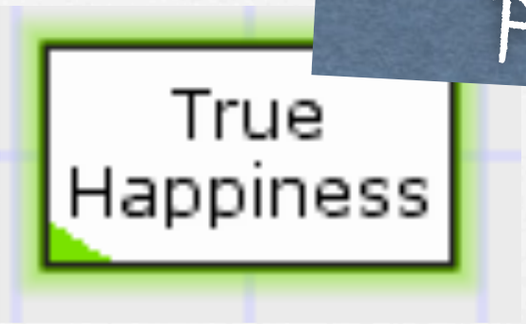
and 

often

Classic Solution

Only try mining constraints for often occurring parameters

- If not 

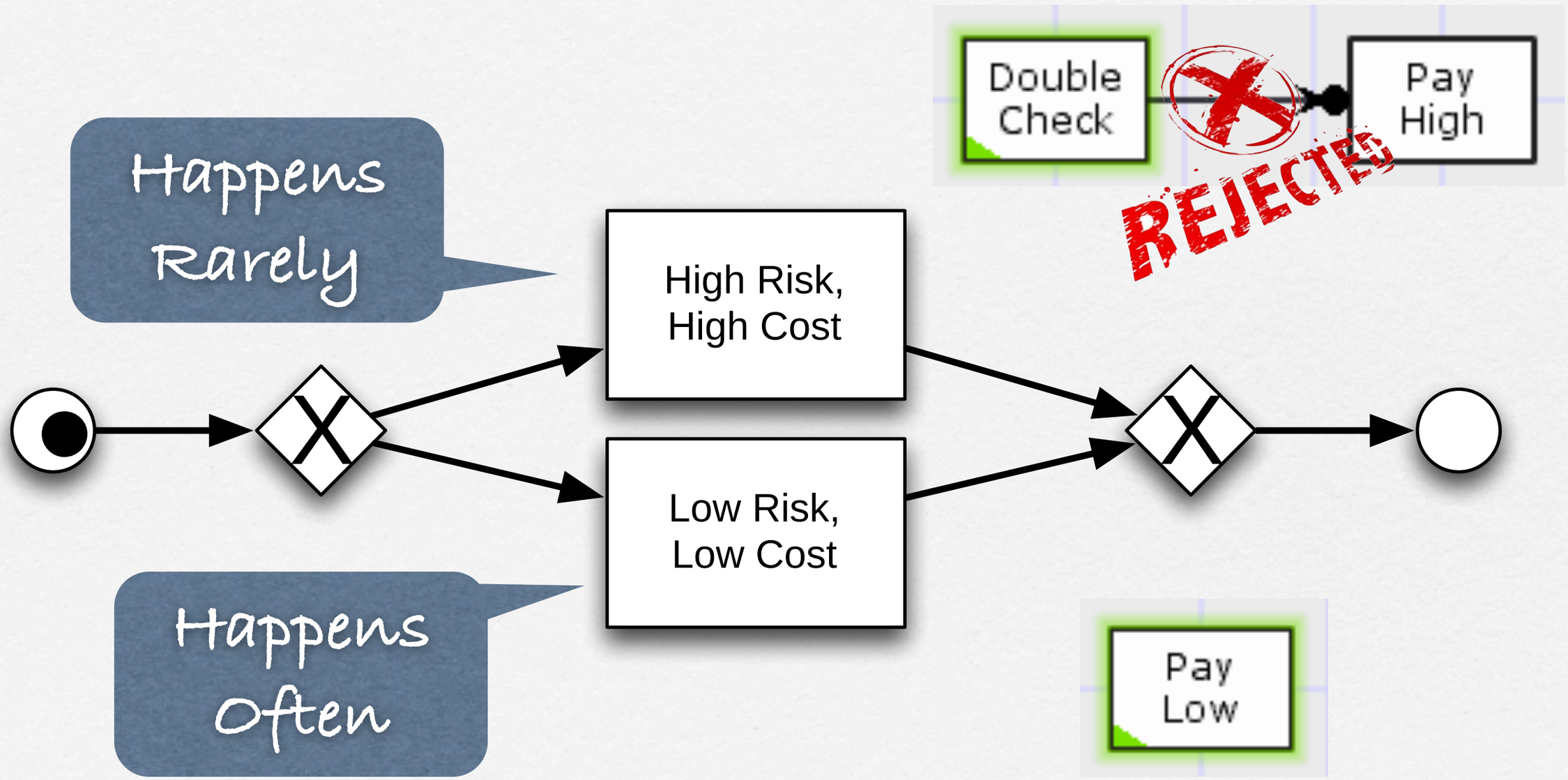
and 

often

- Then not 

often

Problem


Double Check
Pay High
REJECTED
Happens Rarely
High Risk, High Cost
Low Risk, Low Cost
Happens Often
Pay Low

Outline

- Symmetry reduction
- Prefix sharing
- Parallelism
- Superscalarity
- Support
- Use cases

Performance

	Old	MINERful	p
One	129m		2
Easy	550m	26 s	≤ 2
Almost All			≤ 3
All			≤ 5

7.224.024 checks

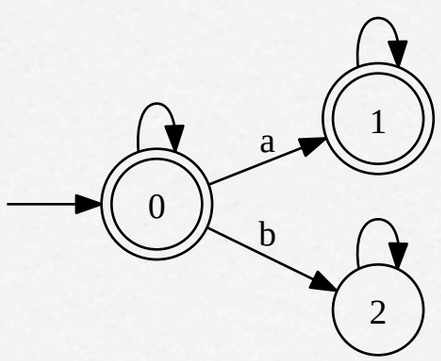
Real data:
13.087 traces
n = 24
24 - 5.100.480 instances

Foundation

Bored, Britney, Happy
Bored, Beer, Happy

1	2	3
1	4	3

- Convert log to efficient structure
- Instead of instantiating constraints, use a single parametrized instance



1	a
2	b
3,4	

1	a
3	b
2,4	

2	a
3	b
1,4	

Performance

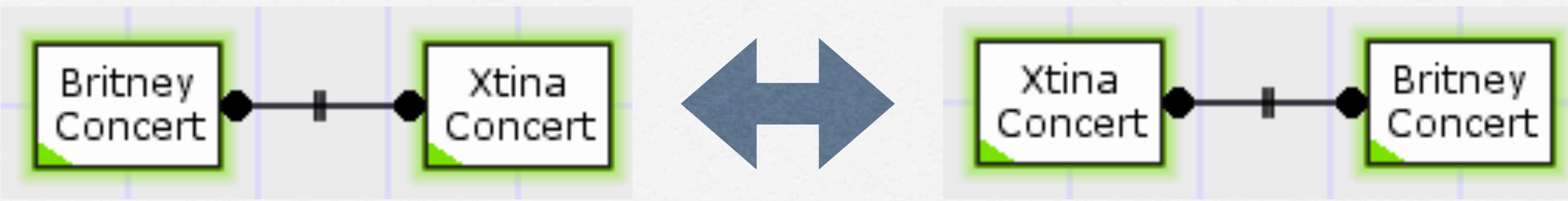
	Old	MINERful	Naïve	p
One	129m			2
Easy	550m	26 s	19 s	≤ 2
Almost All			117 s	≤ 3
All			215 m	≤ 5

7.224.024 checks

158.928.528 checks

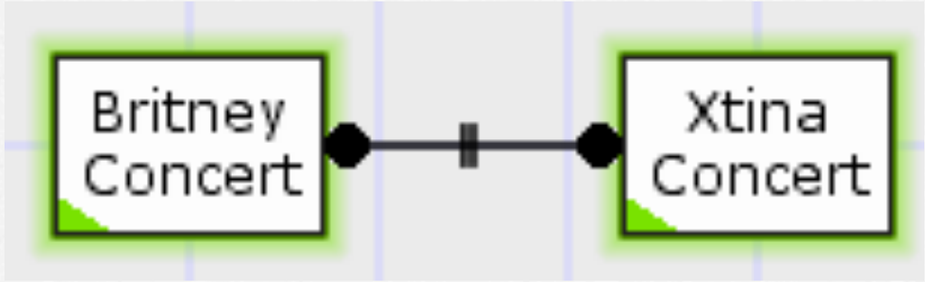
67.749.981.760 checks

Symmetry



- not co-existence(A, B) = not co-existence(B, A)
- Symmetry group: set $S = \{ A_1, A_2, \ldots, A_n \}$ so that any $A_i \in S$ can be swapped for any $A_j \in S$

Symmetry



- Normally, a constraint has $n \cdot (n-1) \cdot \ldots \cdot (n-p+1)$ instantiations
- A symmetry group S means that $|S|!$ of those are the same
- If $|S| = p$, the constraint "only" has $n \cdot (n-1) \cdot \ldots \cdot (n-p+1) / p!$ instantiations
  - This number is also known as $\binom{n}{p}$

Choice 1 of 3:
12.144 instances
158.928.528 checks

Choice 1 of 5:
5.100.480 instances
66.749.981.760 checks

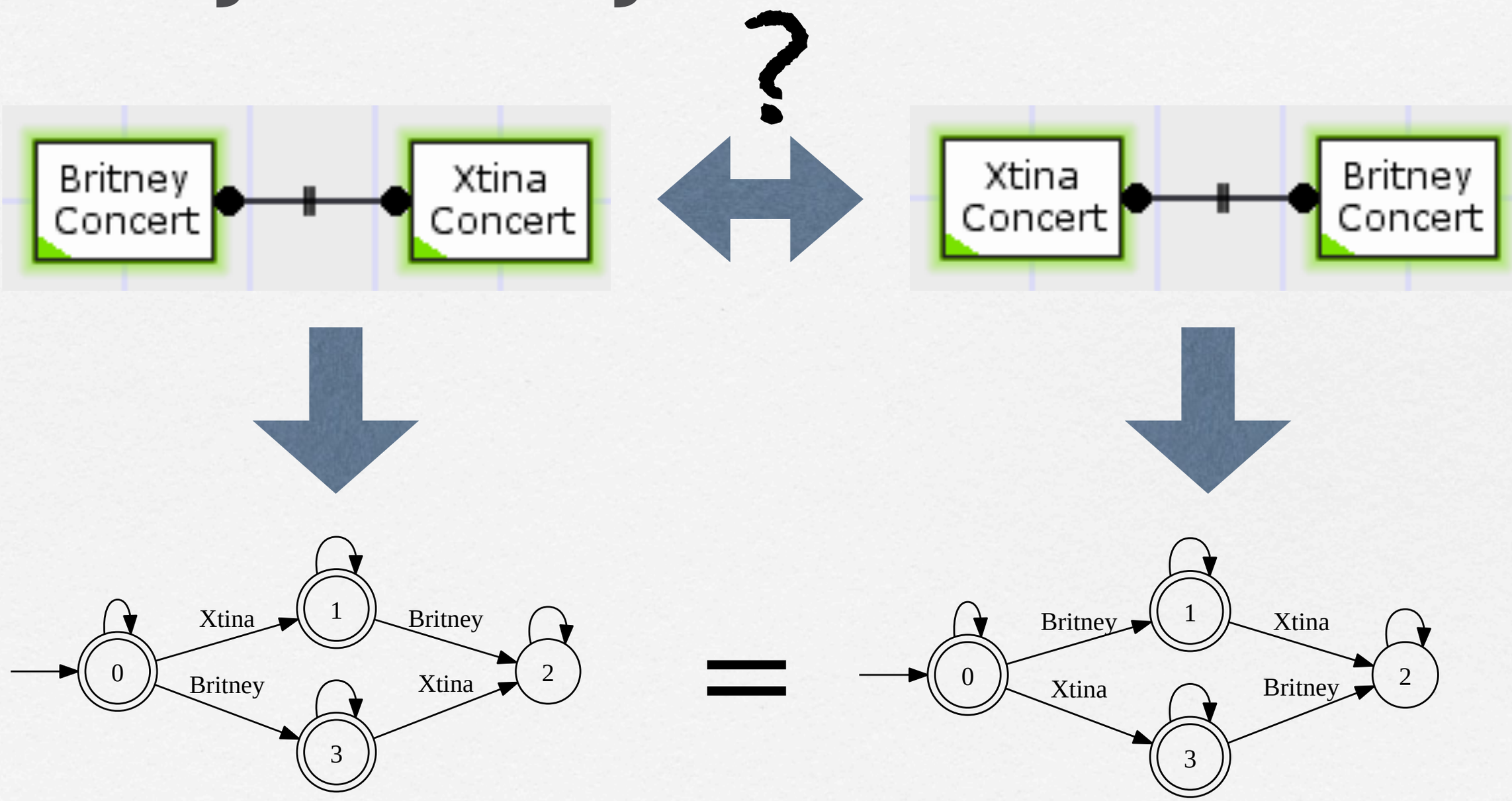
Choice 1 of 3:
3! = 6

Choice 1 of 5:
5! = 120

Choice 1 of 3:
2.024 instances
26.488.088 checks

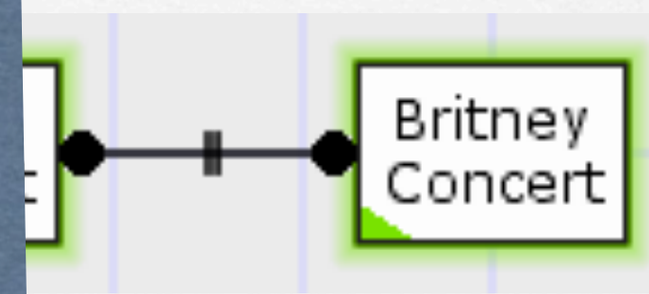
Choice 1 of 5:
42.504 instances
556.249.848 checks

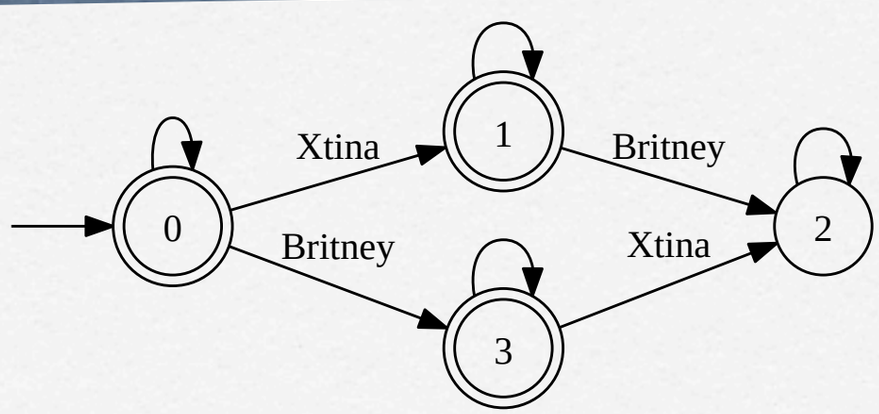
Symmetry


?
Britney Concert
Xtina Concert
Xtina Concert
Britney Concert
0
1
2
3
Xtina
Britney
Britney
Xtina
=
0
1
2
3
Britney
Xtina
Xtina
Britney

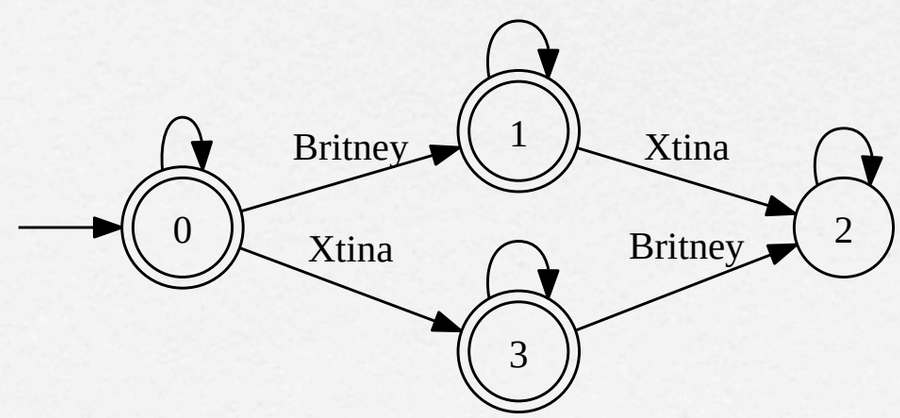
Symmetry

Here we mean whether $L(A_1) = L(A_2)$
We check this by checking that
$L(A_1) \setminus L(A_2) = L(A_1) \cap L(A_2)^c =$
$L(A_1) \cdot L(A_2)^c = \emptyset$ and
$L(A_2) \cdot L(A_1)^c = \emptyset$





=



Performance

	Old	MINERful	Naïve	Sym		
One	129m				2	
Easy	550m	26 s	19 s	16 s	≤ 2	2 s
Almost All			117 s	35 s	≤ 3	7 s
All			215 m	154 s	≤ 5	56 s

~~7.224.024~~
3.612.012
checks

~~66.749.981.760~~
556.249.848
checks

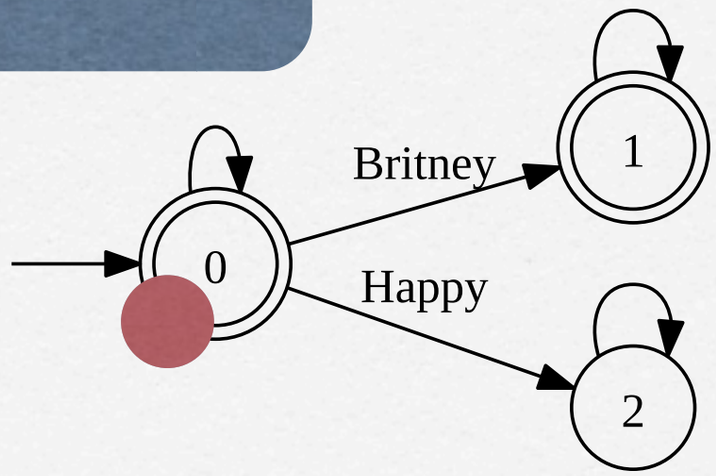
~~158.928.528~~
26.488.088 checks

Prefix Sharing

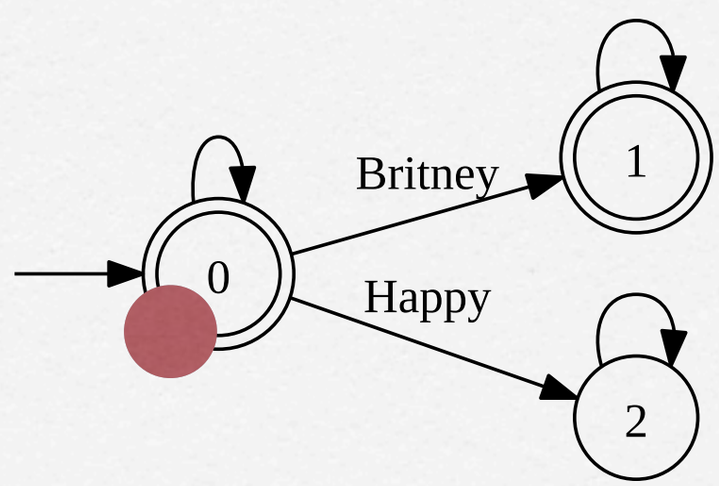
Bored, Britney, Happy

Bored, Beer, Happy

We replay the "Bored" transition twice...



Prefix Sharing



Performance

	Old	MINERful	Naïve	Sym	Prefix	Super
One	129m					
Easy	550m	26 s	19 s	16 s	43 s	2 s
Almost All			s	35 s	112 s	7 s
All			m	154 s	437 s	56 s

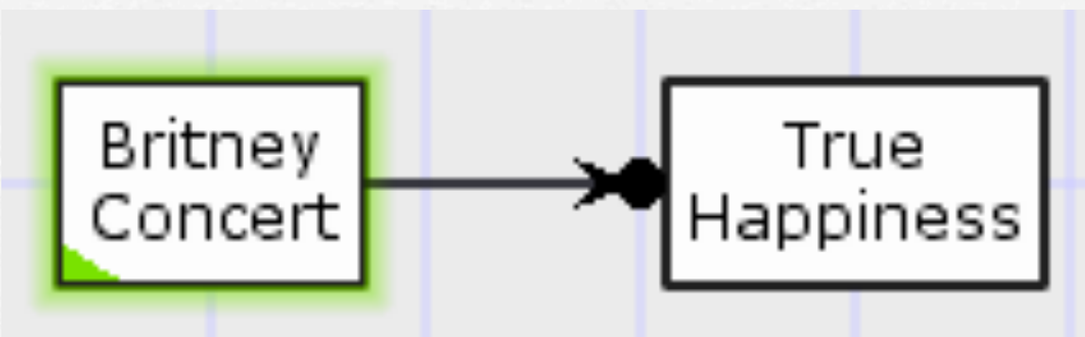
While we do get sharing (factor 4), checking becomes more expensive

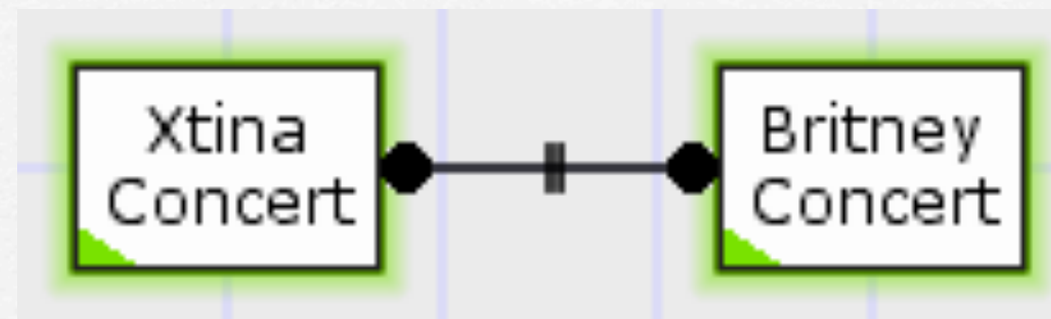
Parallelization

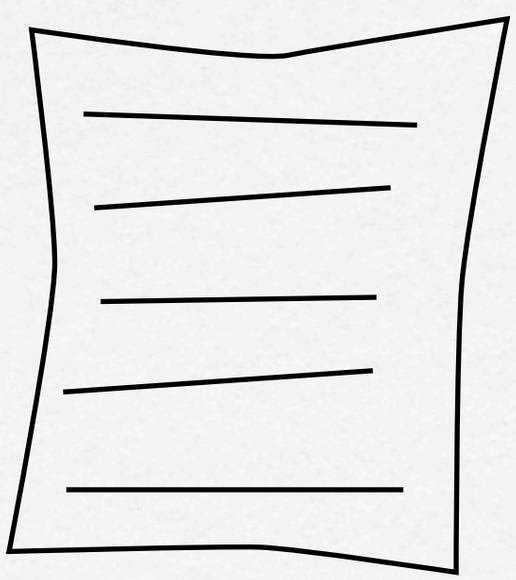
Mine each constraint on each core

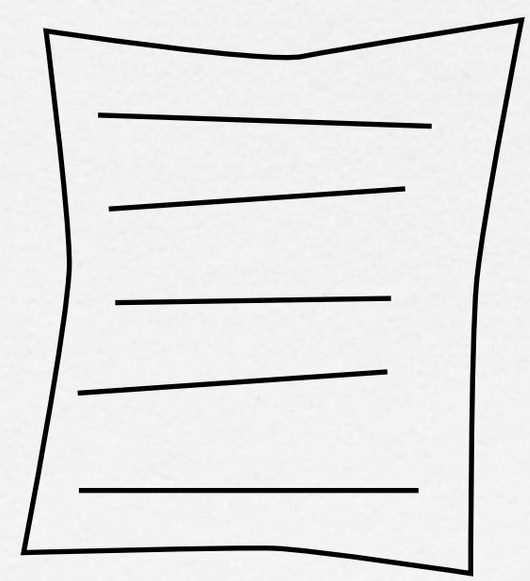
Split log and mine each fragment on each core

Parallelization

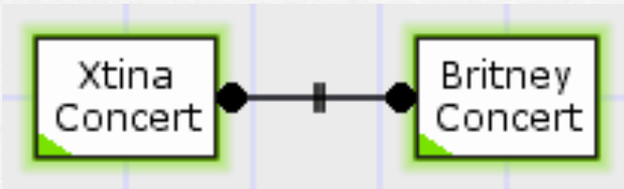


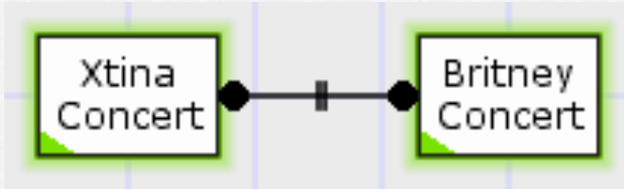


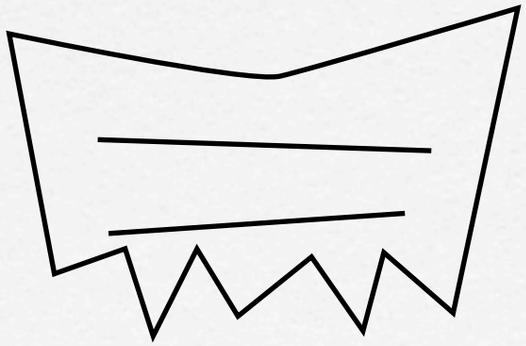


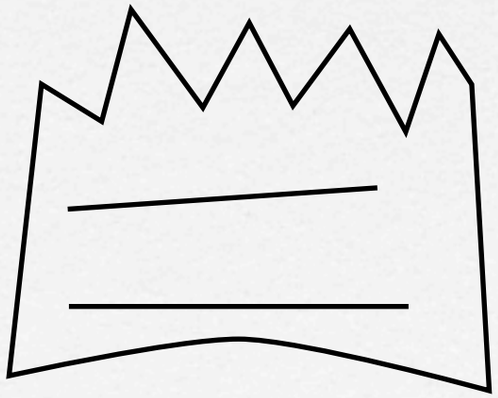


Parallelization

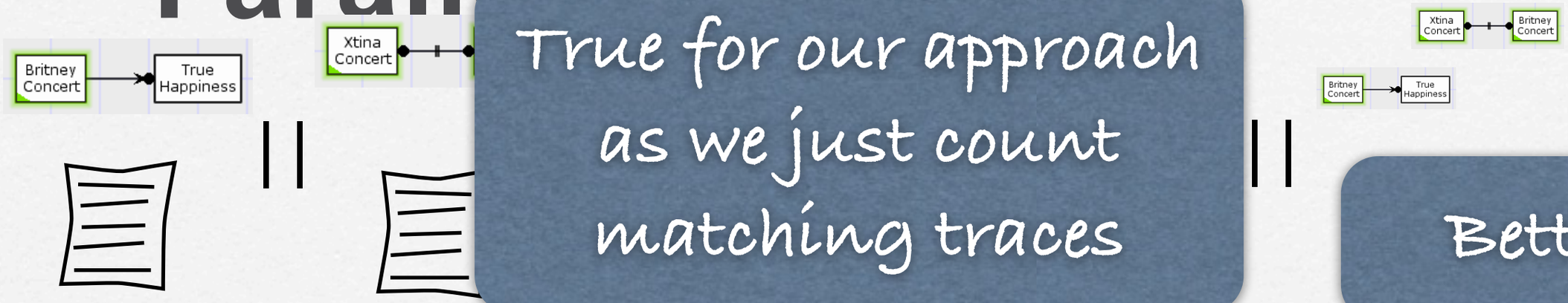








Parallelization



True for our approach as we just count matching traces

Better

Mine constraints in parallel	Mine traces in parallel
Good for many constraints	Good for many traces
Assumes constraints are similar	Assumes traces are similar
Requires models are composable	Requires results are composable

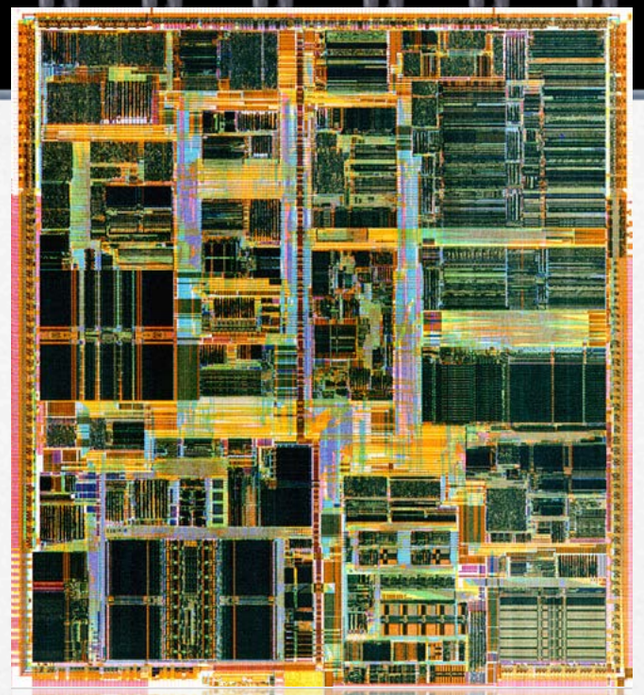
REJECTED

Performance

Just 2 computation threads

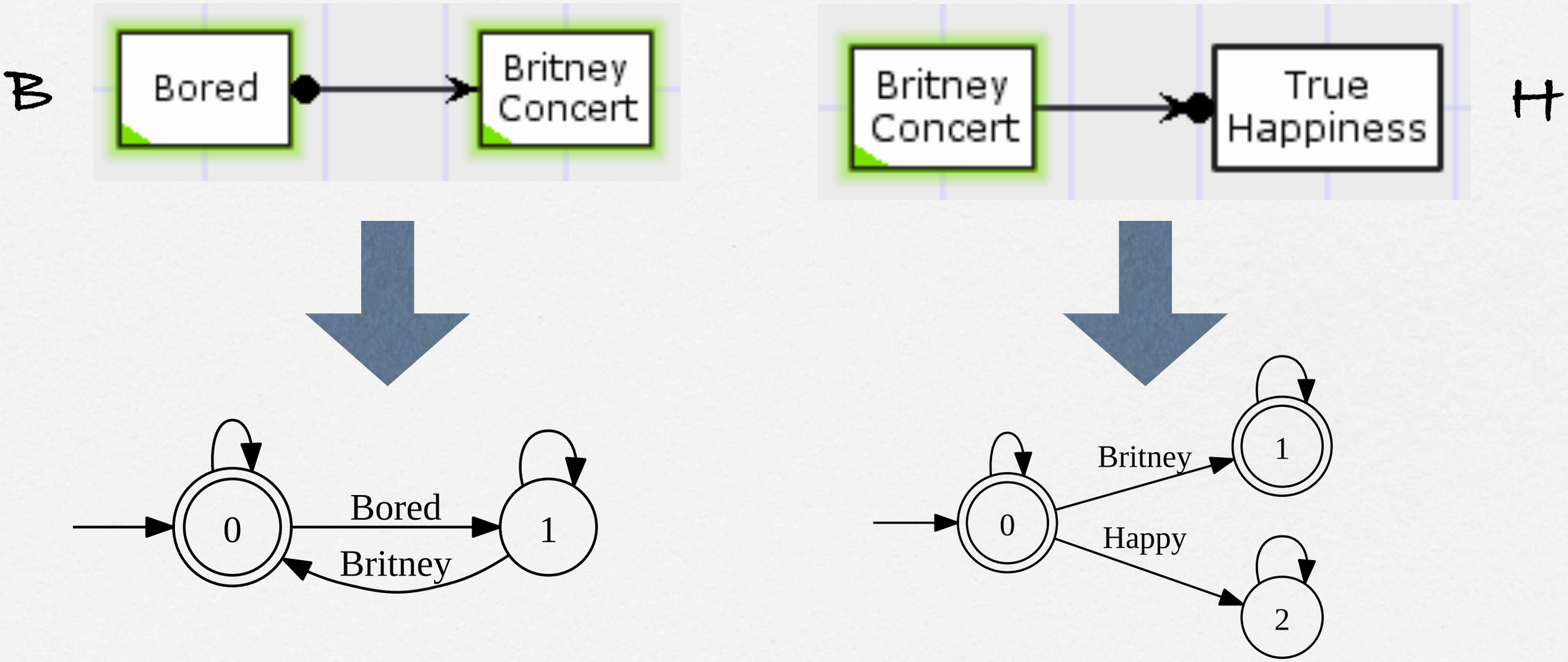
	Old	MINERful	Naïve	Sym	Par	Super
One	129m					
Easy	550m	26 s	19 s	16 s	12 s	2 s
Almost All			117 s	35 s	27 s	7 s
All			215 m	154 s	108 s	56 s

Superscalar Mining

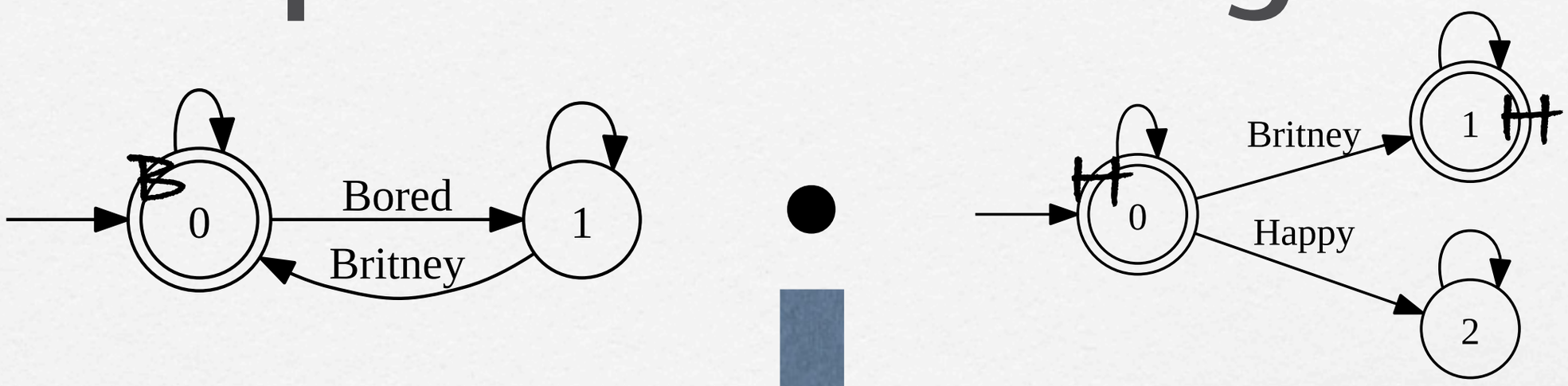


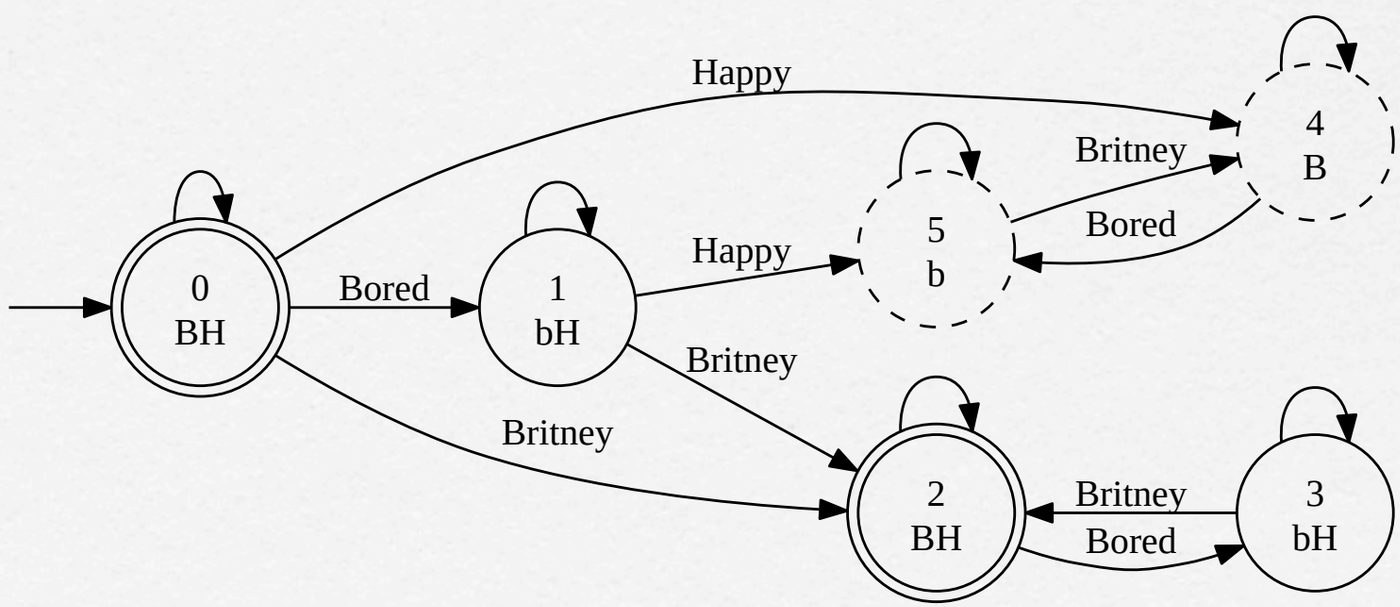
- Superscalar CPUs (90s): improve speed by executing multiple instructions at once
- Superscalar mining: improve speed by mining multiple constraints at once

Superscalar Mining



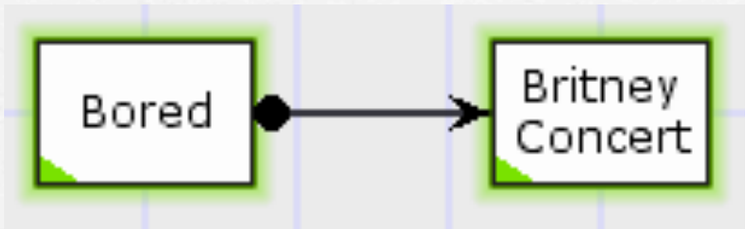
Superscalar Mining

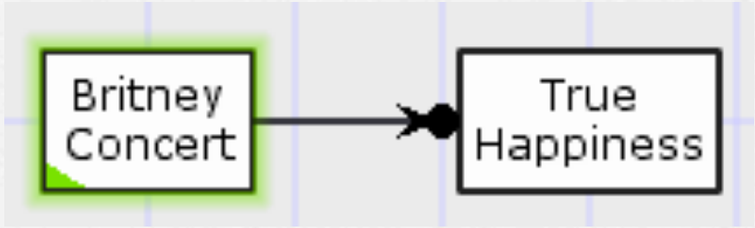


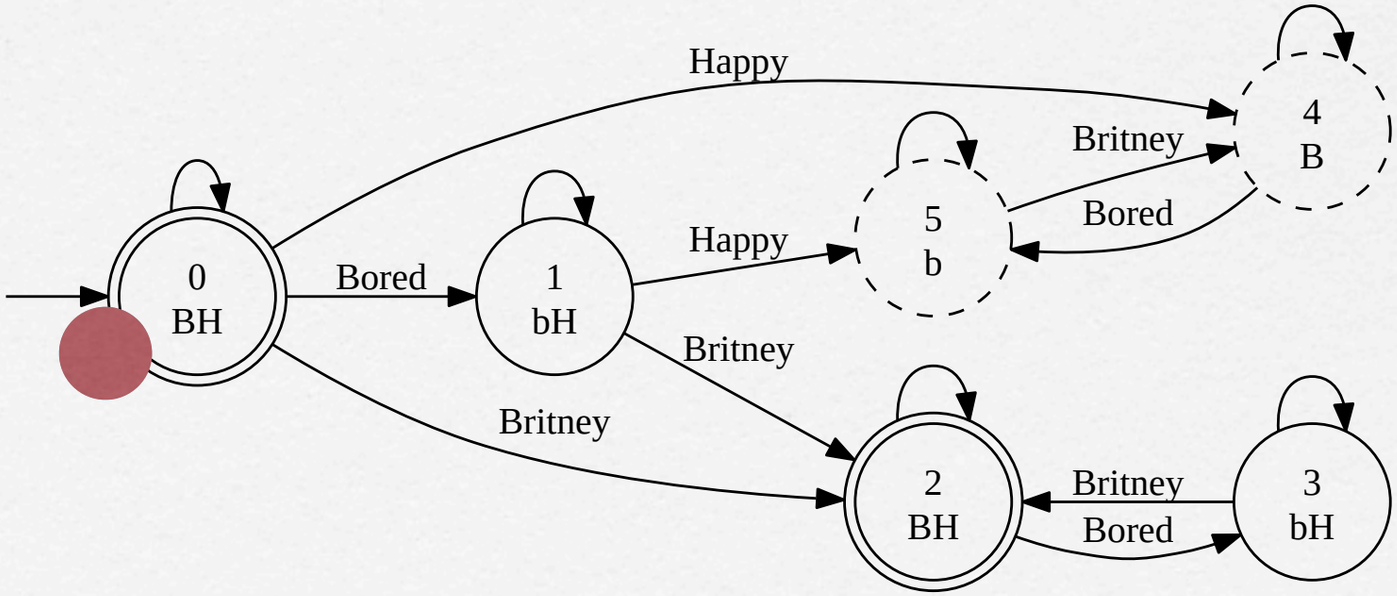


Superscalar Mining

Bored, Britney, Happy







Superscalar Mining

- We do not want to split up symmetry groups (speed)
- We do not want to make the automaton too large (memory)

Superscalar Mining

- Given two sets of constraints, A and B
- A can be merged into B if
  - We do not break any symmetry groups doing so
  - All constraints in A are special cases of all constraints in B

lining

for all S_A of A there is a S_B of B such that $S_A \subseteq S_B$, and for all S of A with $S \cap S_B \neq \emptyset$, we have $S = S_A$

for all C_A of A there is a C_B of B so that $C_A \Rightarrow C_B$

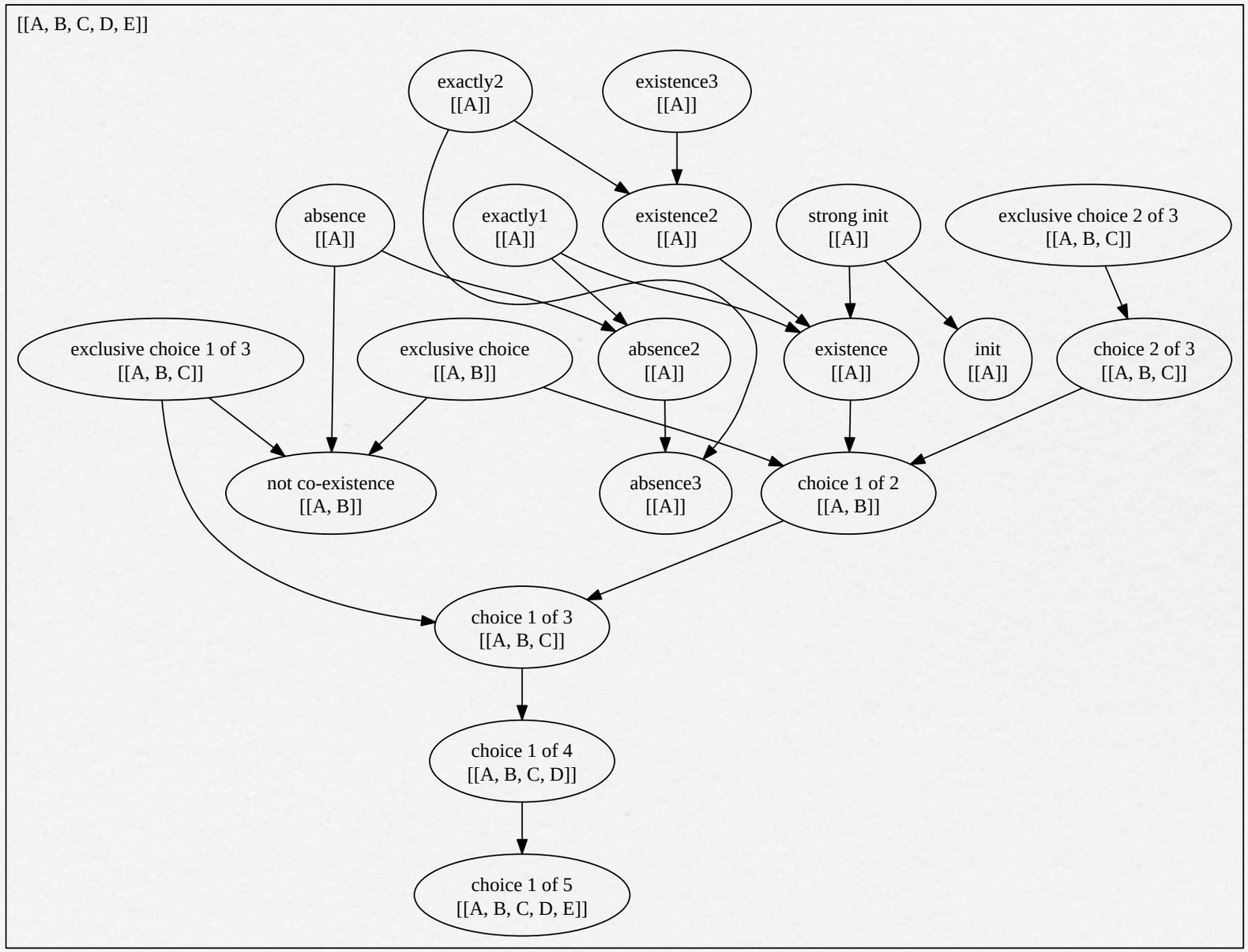
- A can be merged into B if

- We d... symmetry groups

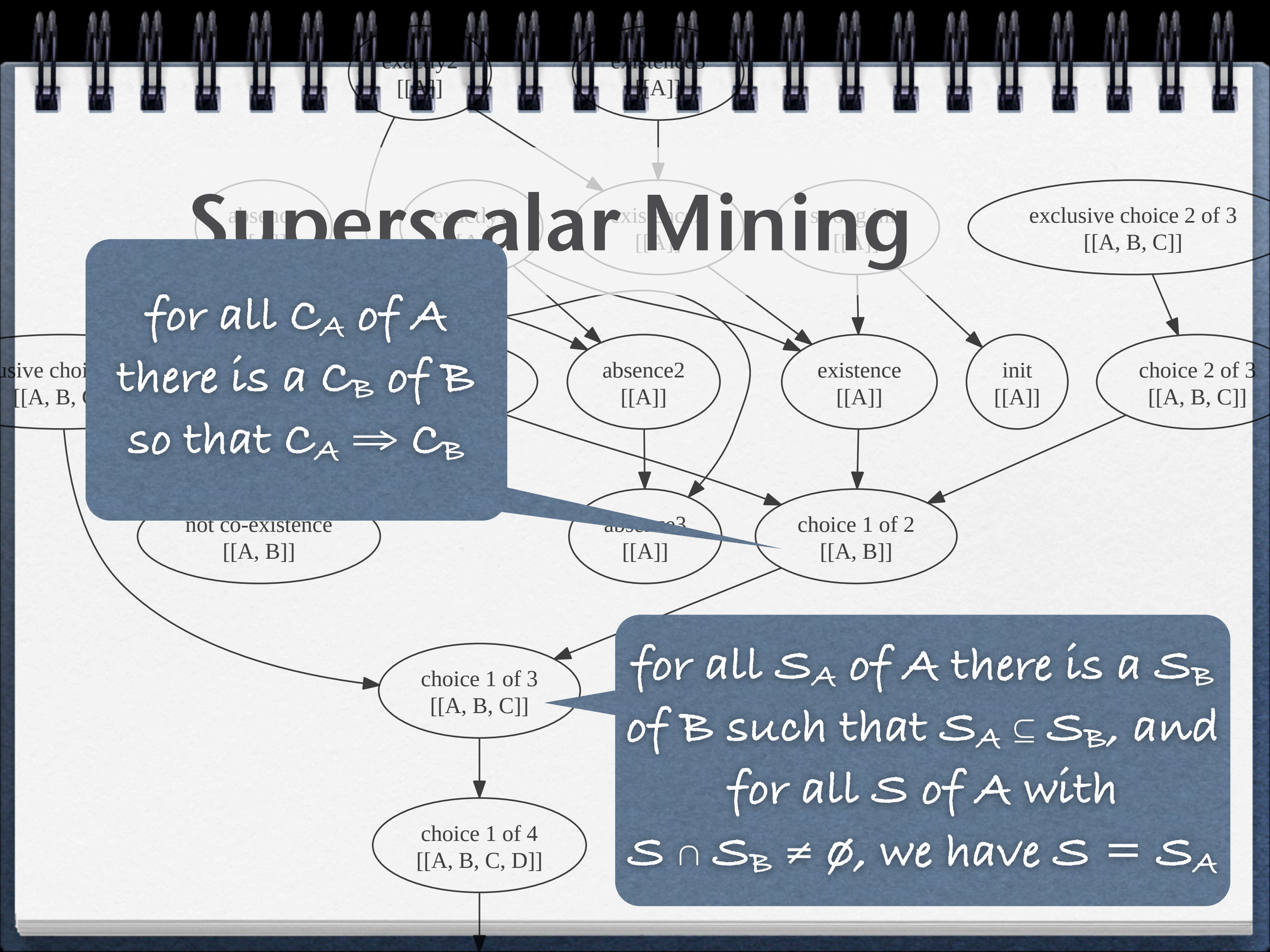
<u>Note</u>: This is not symmetrical! It captures that it is ok to add something easy to something hard, but not the other way around

...re special cases of

Superscalar Mining



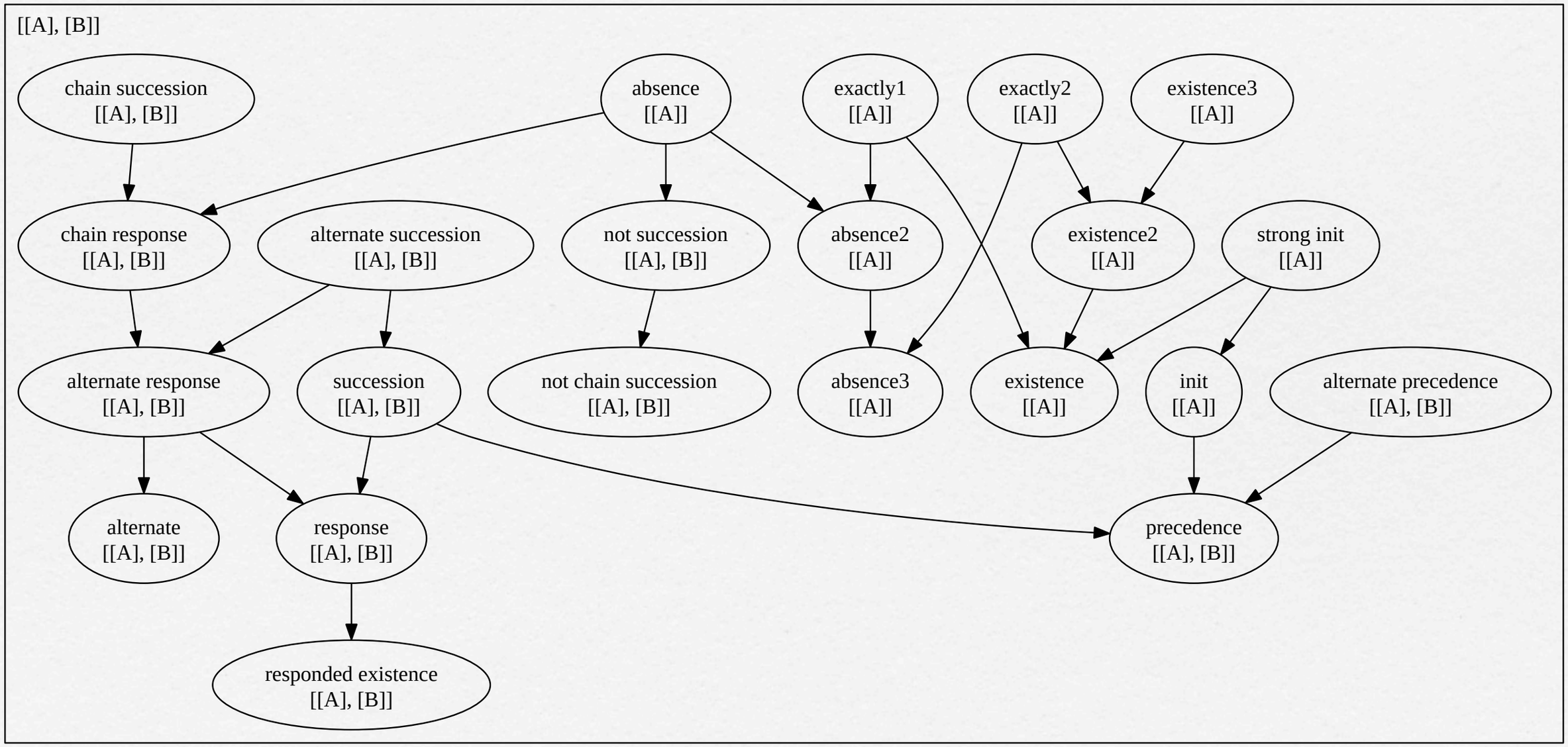
Superscalar Mining



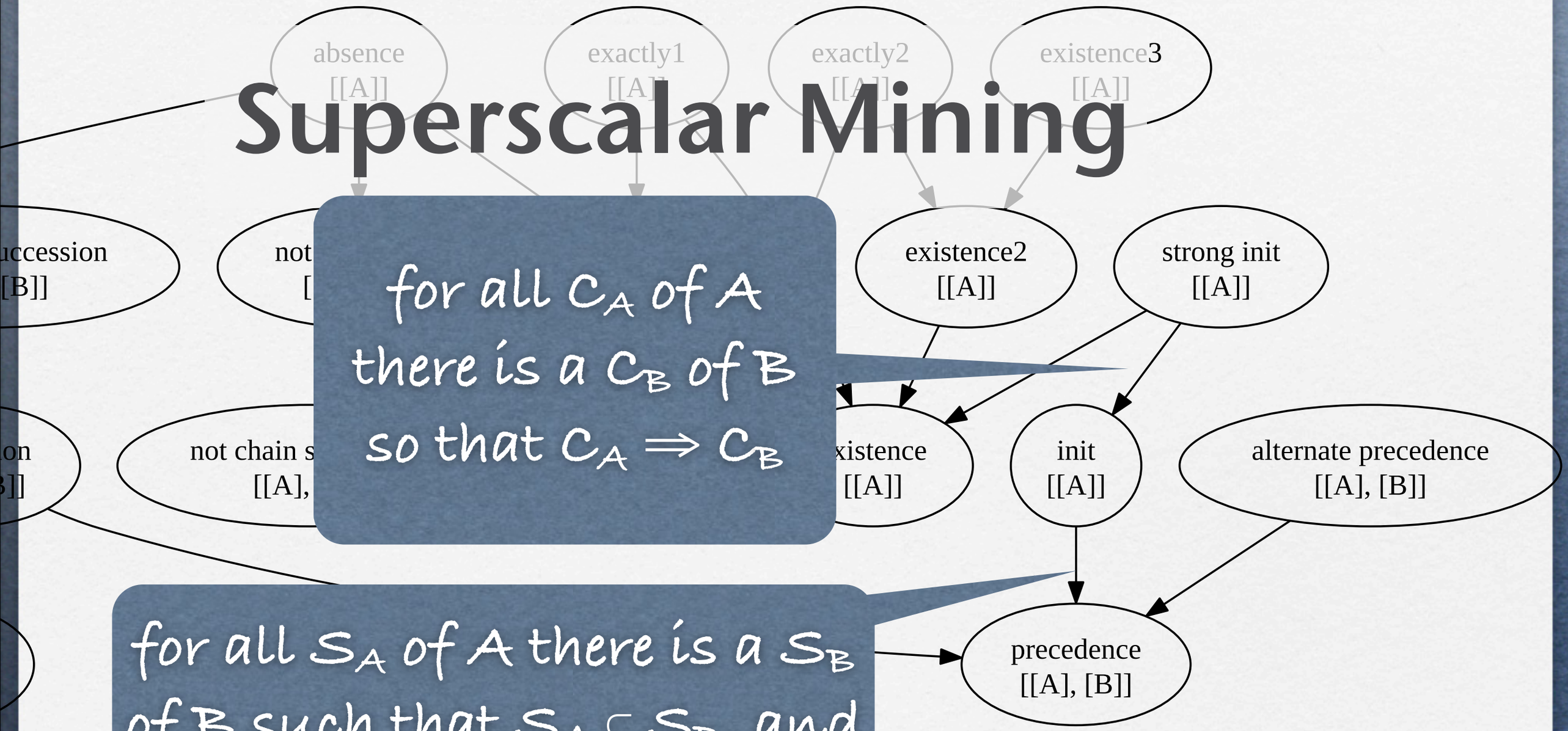
for all c_A of A there is a c_B of B so that $c_A \Rightarrow c_B$

for all s_A of A there is a s_B of B such that $s_A \subseteq s_B$, and for all s of A with $s \cap s_B \neq \emptyset$, we have $s = s_A$

Superscalar Mining



Superscalar Mining



for all c_A of A there is a c_B of B so that $c_A \Rightarrow c_B$

for all s_A of A there is a s_B of B such that $s_A \subseteq s_B$, and for all s of A with $s \cap s_B \neq \emptyset$, we have $s = s_A$

Superscalar Mining

- We can check 34 Declare constraints using just 3 checks:
  - 20 "choices": 53 states, SG {{A, B, C, D, E}}
  - 22 "orders": 112 states, SG {{A}, {B}}
  - 2 "outliers": 7 states, SG {{A}, {B}}

Performance

Using 8 (slower) CPUs, we get below 30 seconds

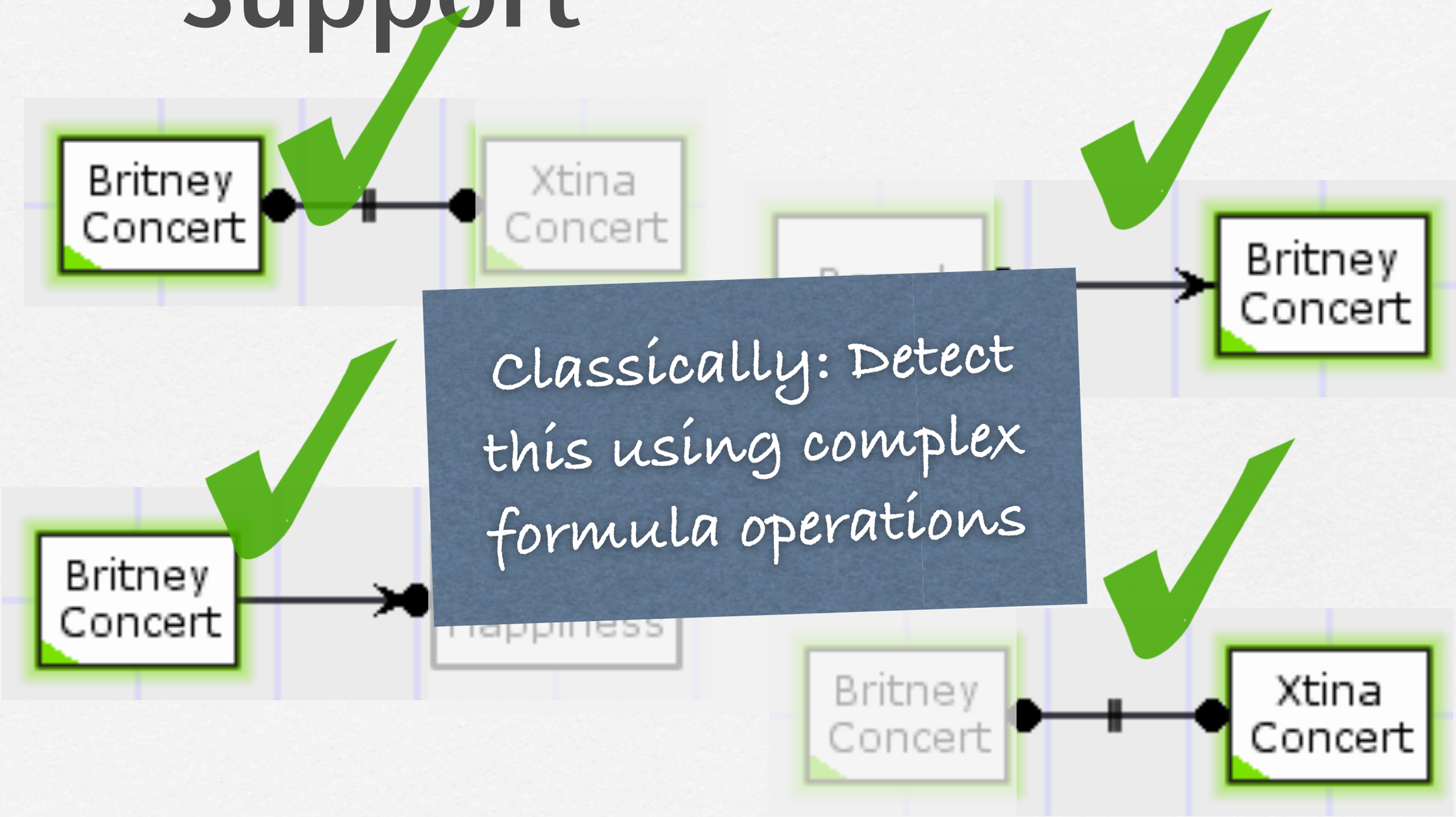
	Old	MINERful	Naïve	Sym	Par	Super
One	129m					
Easy	550m	26 s	19 s	16 s	12 s	2 s
Almost All			117 s	35 s	27 s	7 s
All			215 m	154 s	108 s	56 s

Perf

If the old ProM miner has as good progression as our naïve miner, it would take just under 259 days to mine all constraints; if it's as good as our best, it would take more than 10 days

	Old					
One	129m					
Easy	550m	26 s	19 s	16 s	12 s	2 s
Almost All			117 s	35 s	27 s	7 s
All			215 m	154 s	108 s	56 s

Support

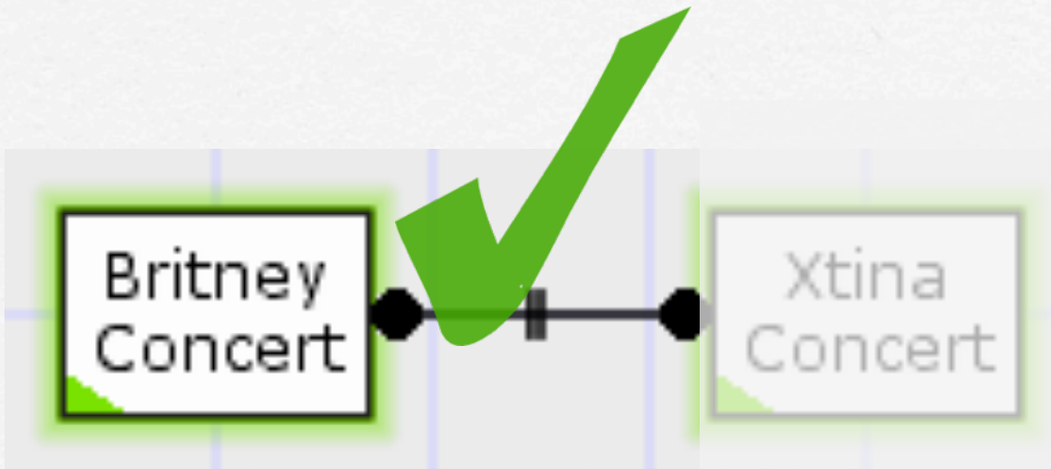


Classically: Detect this using complex formula operations

Support

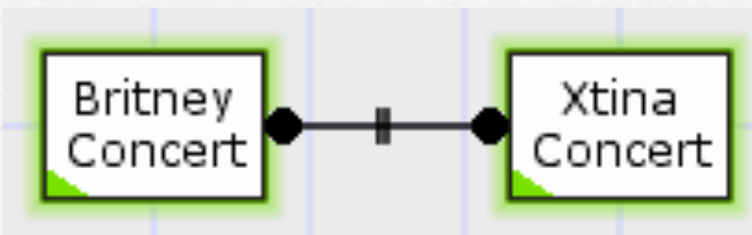
Classically: Detect this using complex formula operations

Declare doesn't have non-trivial sub-behavior, so this just corresponds to preventing a task



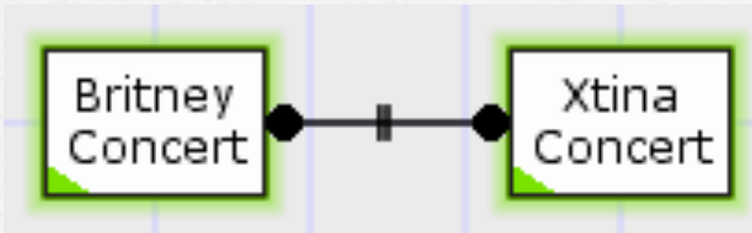
The idea of the formula operations is to prevent certain sub-behavior and see if the constraint becomes trivially true

Support



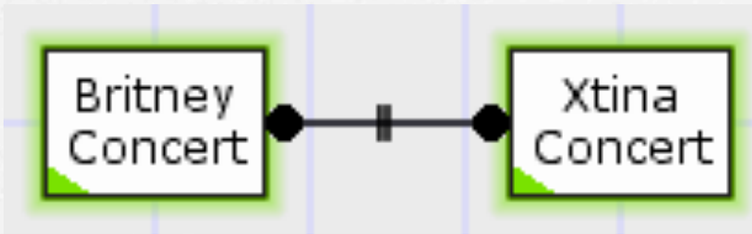
ξ

= true?

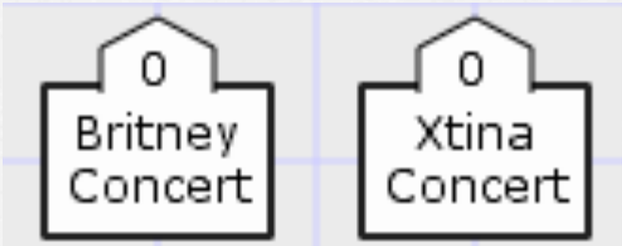


ξ

= true?

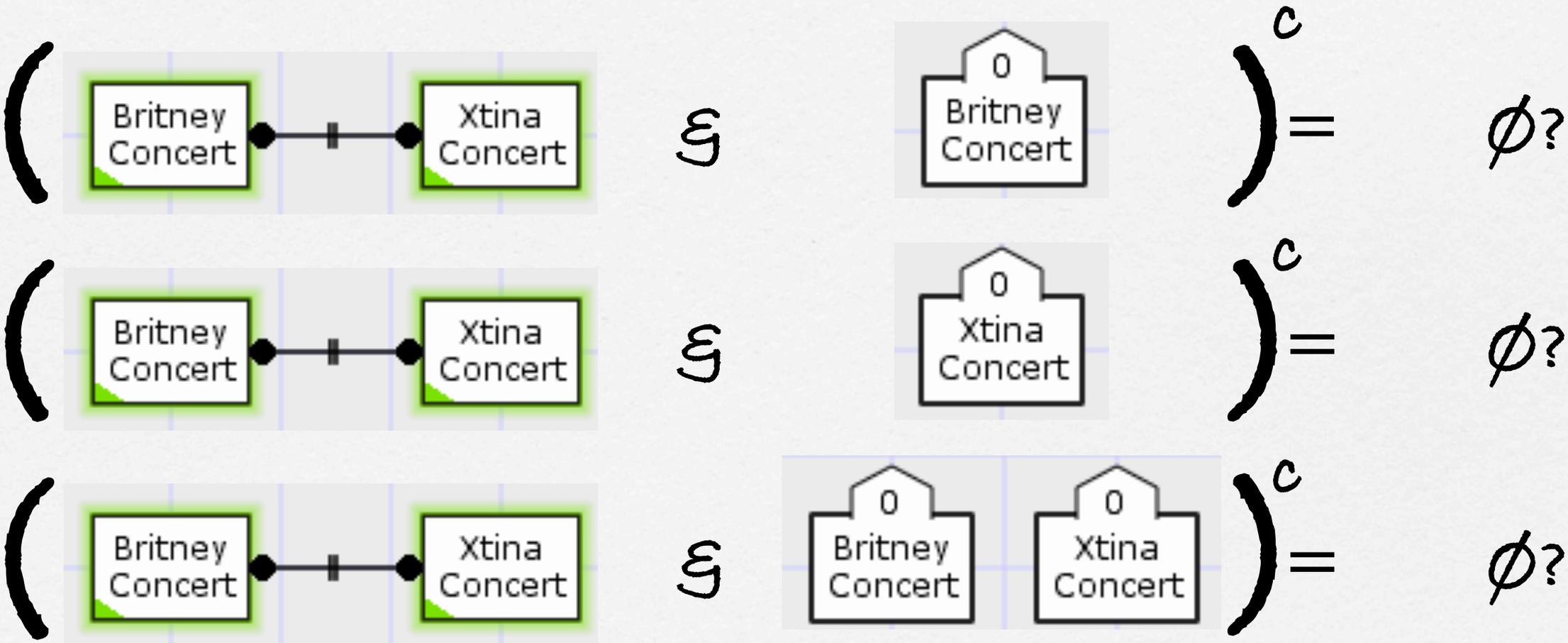


ξ

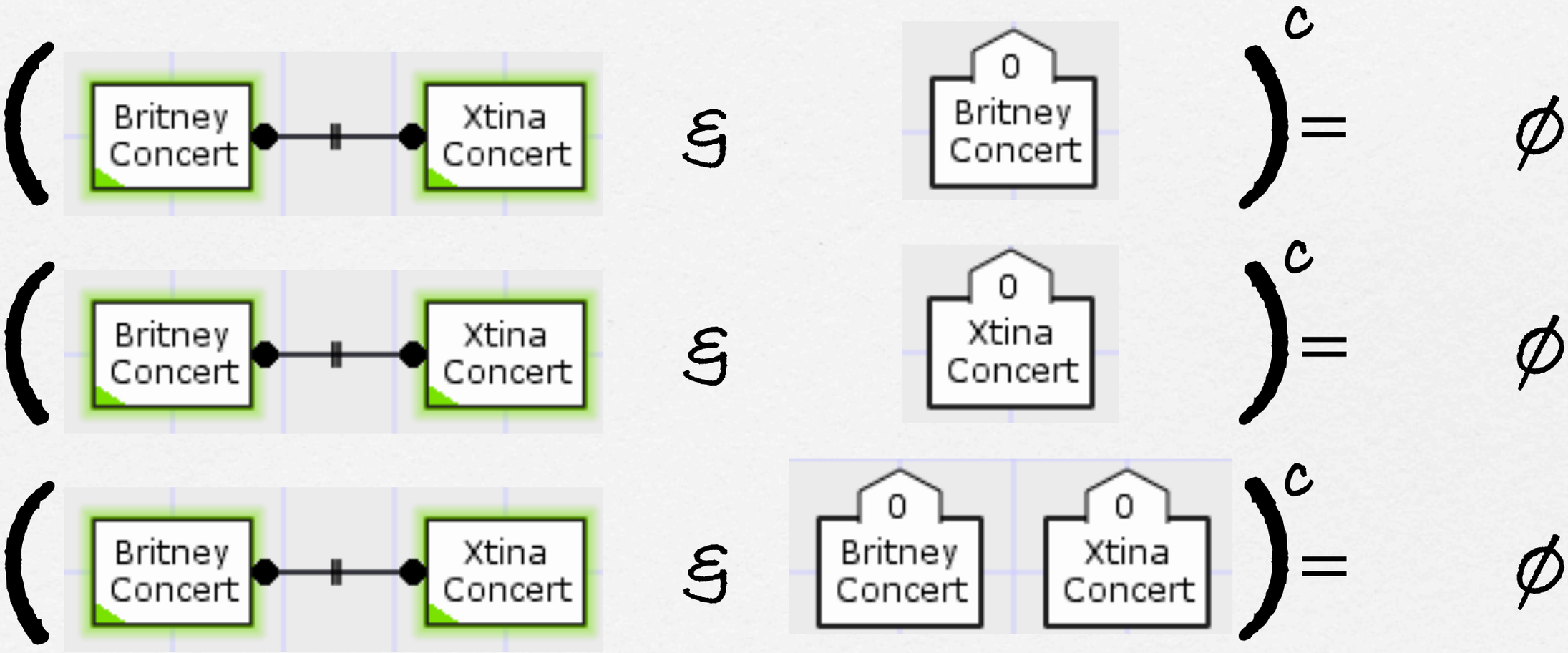


= true?

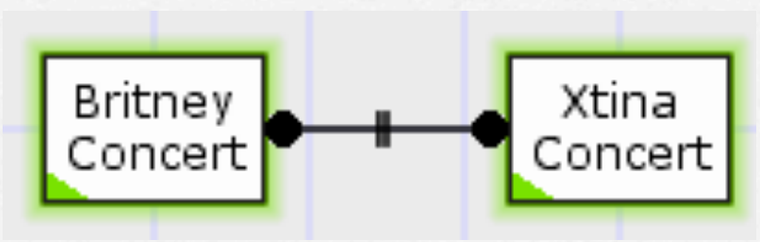
Support


Britney Concert
Xtina Concert
0
Britney Concert
0
Xtina Concert
0
Britney Concert
0
Xtina Concert

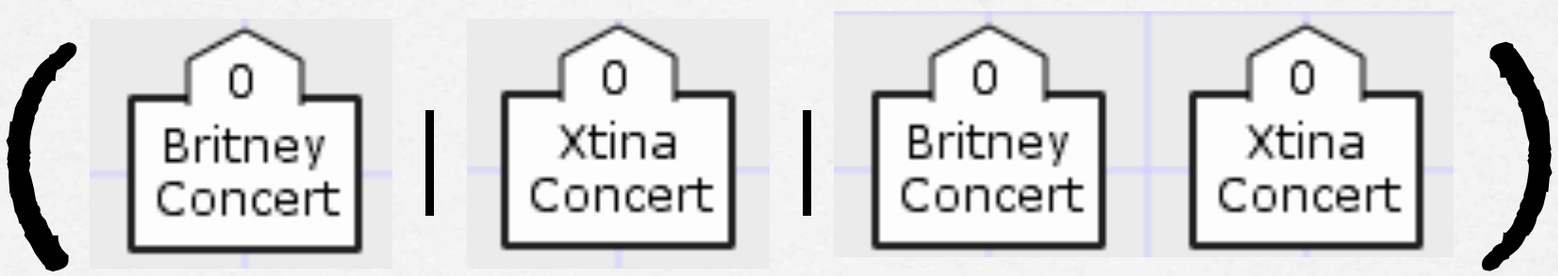
Support


Britney Concert
Xtina Concert
0
Britney Concert
0
Xtina Concert
0
Britney Concert
0
Xtina Concert

Support

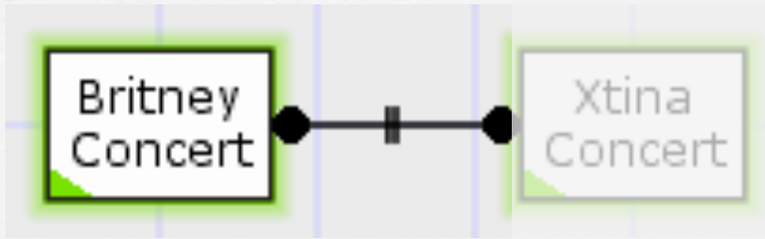


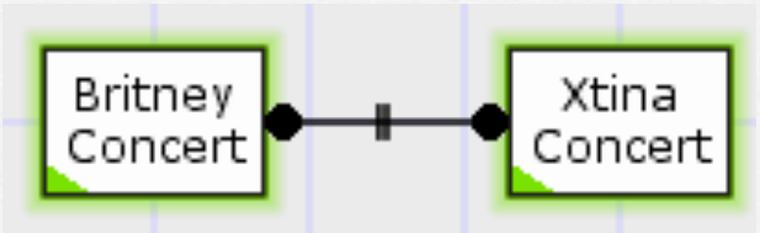
ξ



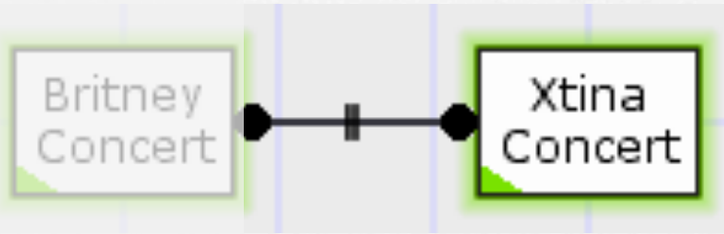
= true

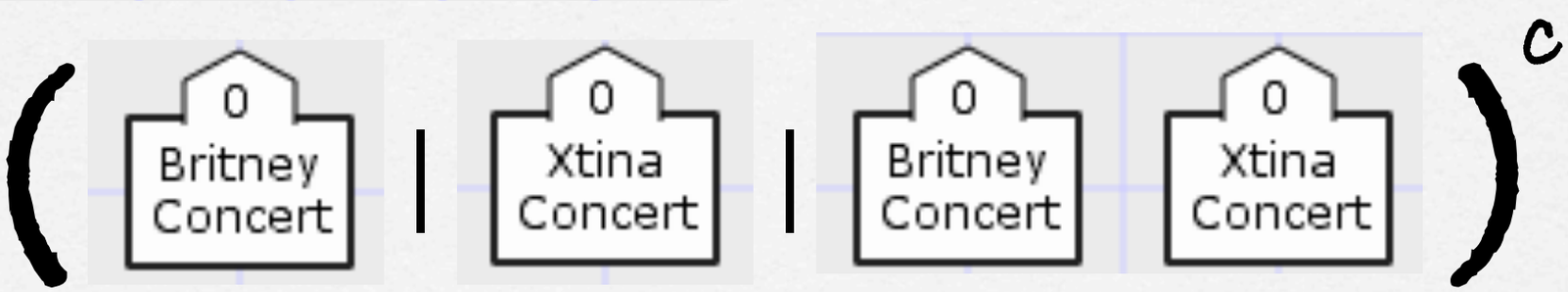
Support





ξ

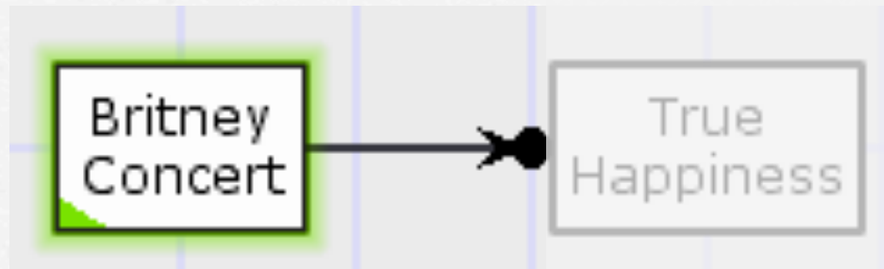


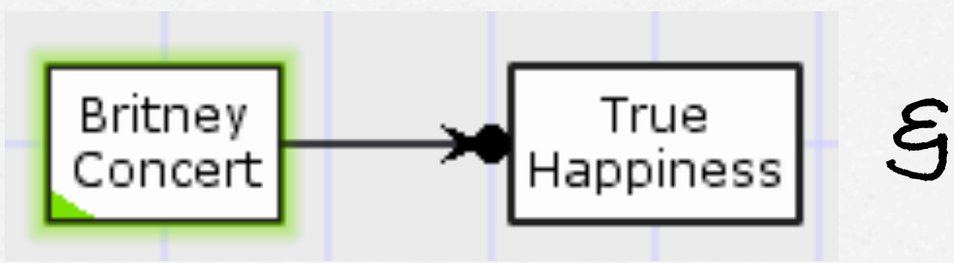


^{c} $=$?

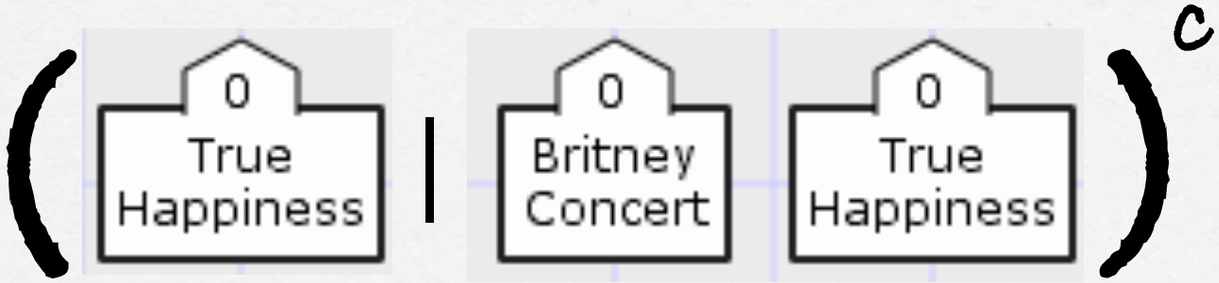
This expression is our measurement of support for the not co-existence constraint

Support





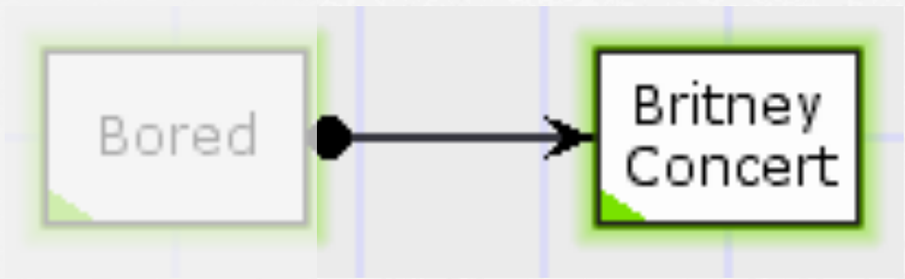
ξ



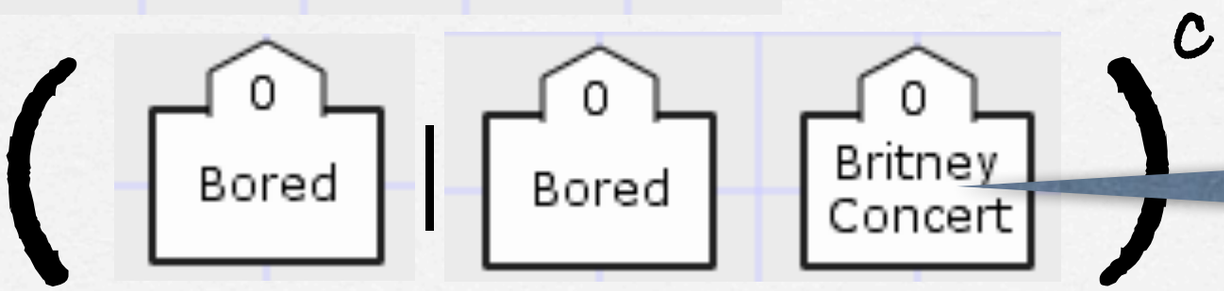
^{c} $=$?

This expression is our measurement of support for the precedence constraint

Support



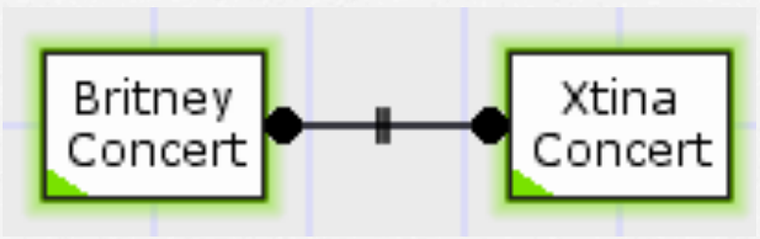
ξ



The reason for having exclusions of more than one constraint is disjunctive splits

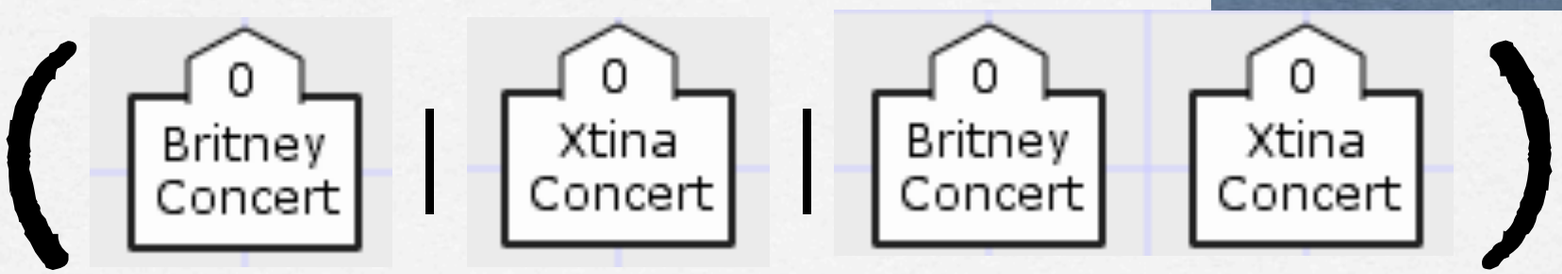
This expression is our measurement of support for the response constraint

Support

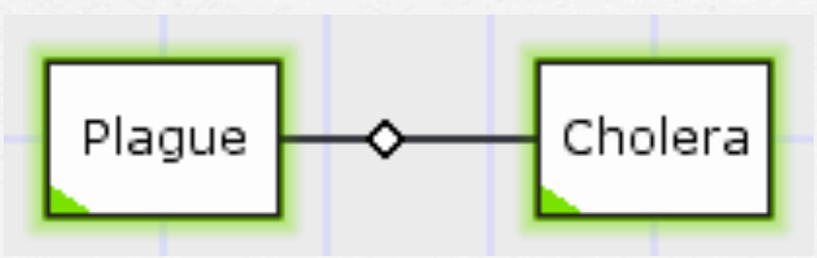


ξ

Conjunction with the support expression is always true $\Rightarrow$ positive support

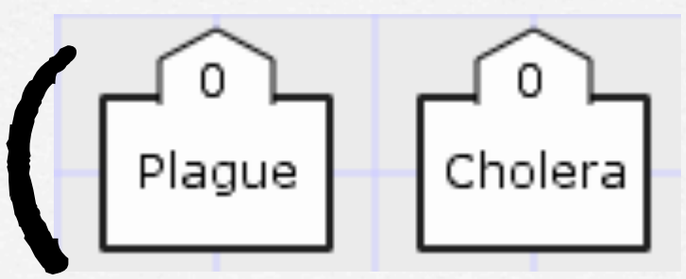


$= \quad \text{true}$



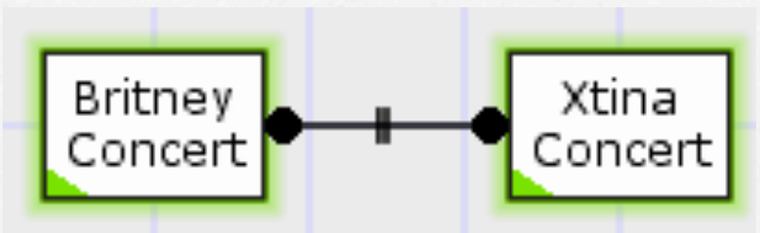
ξ

Conjunction with the support expression is always false $\Rightarrow$ negative support



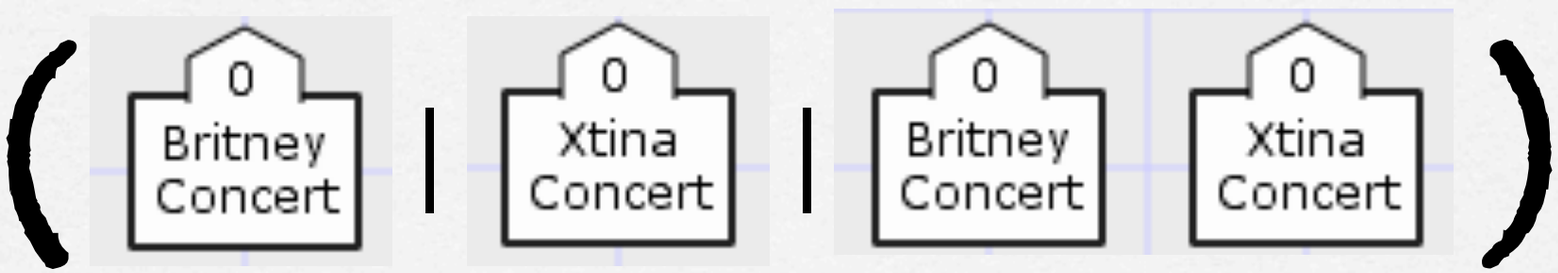
$= \quad \text{false}$

Support

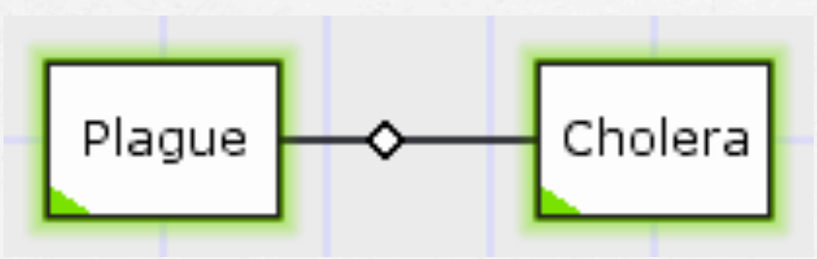


ξ

"Holds against all odds"

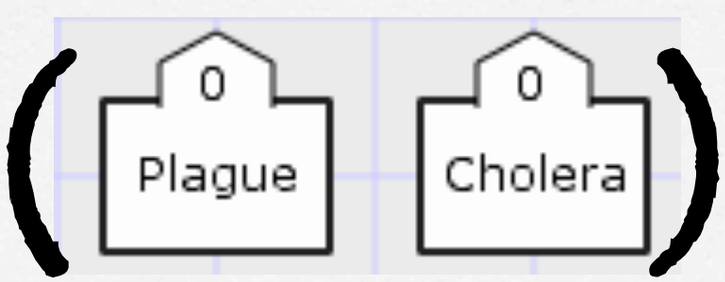


= true



ξ

"Never had a chance to hold"



This is by the way very related to apriori reduction

= false

Support

- Our notion of support coincides with previous
- Previous notion and ours induce a notion of confidence that is identical when previous is defined
- Our is computed automatically and works for all constraints

Comparison

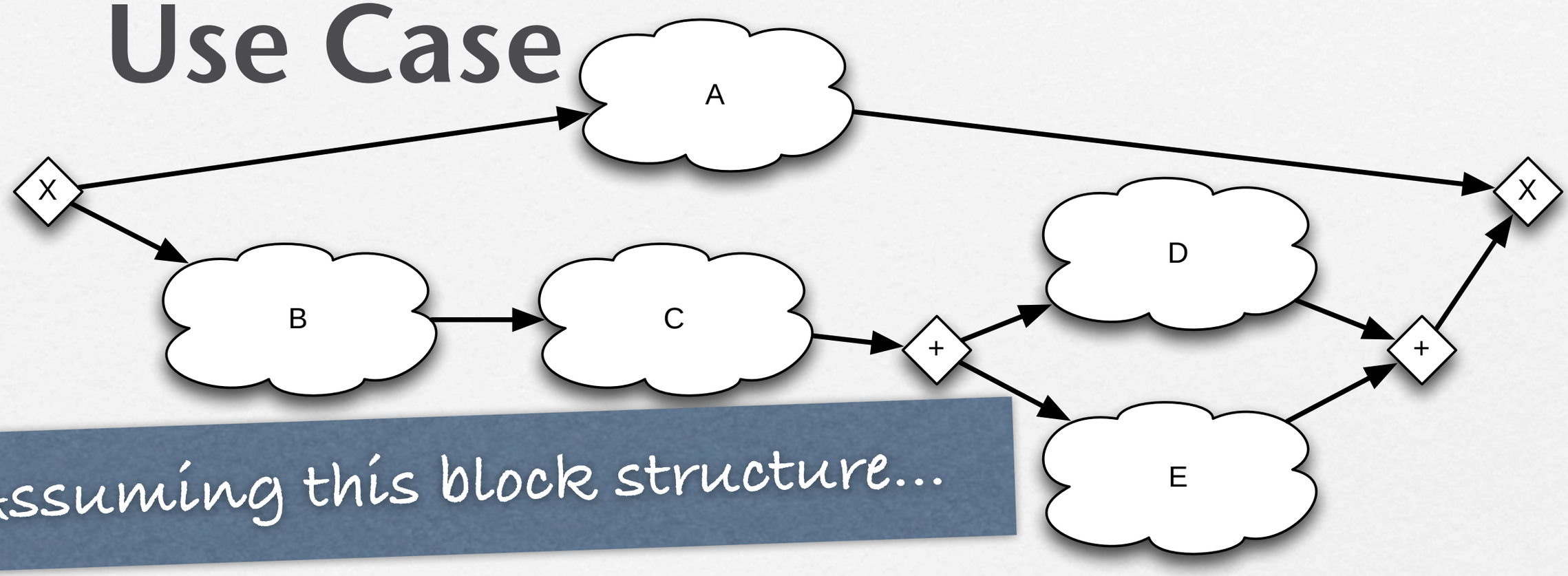
	ProM Miner	MINERful	Unconstrained Miner
Speed	Slow!!	Fast	Faster
Extensibility	Good	None	Better
Confidence choices	Good	Good	Medium
Confidence generality	None	None	Full

Comparison

	ProM Miner		
Speed	Slow!!		
Extensibility	Good	None	Better
Confidence choices	Good	Good	Medium
Confidence generality	None	None	Full

The reason is that some choices of confidence do not make sense in general

Use Case



Assuming this block structure...

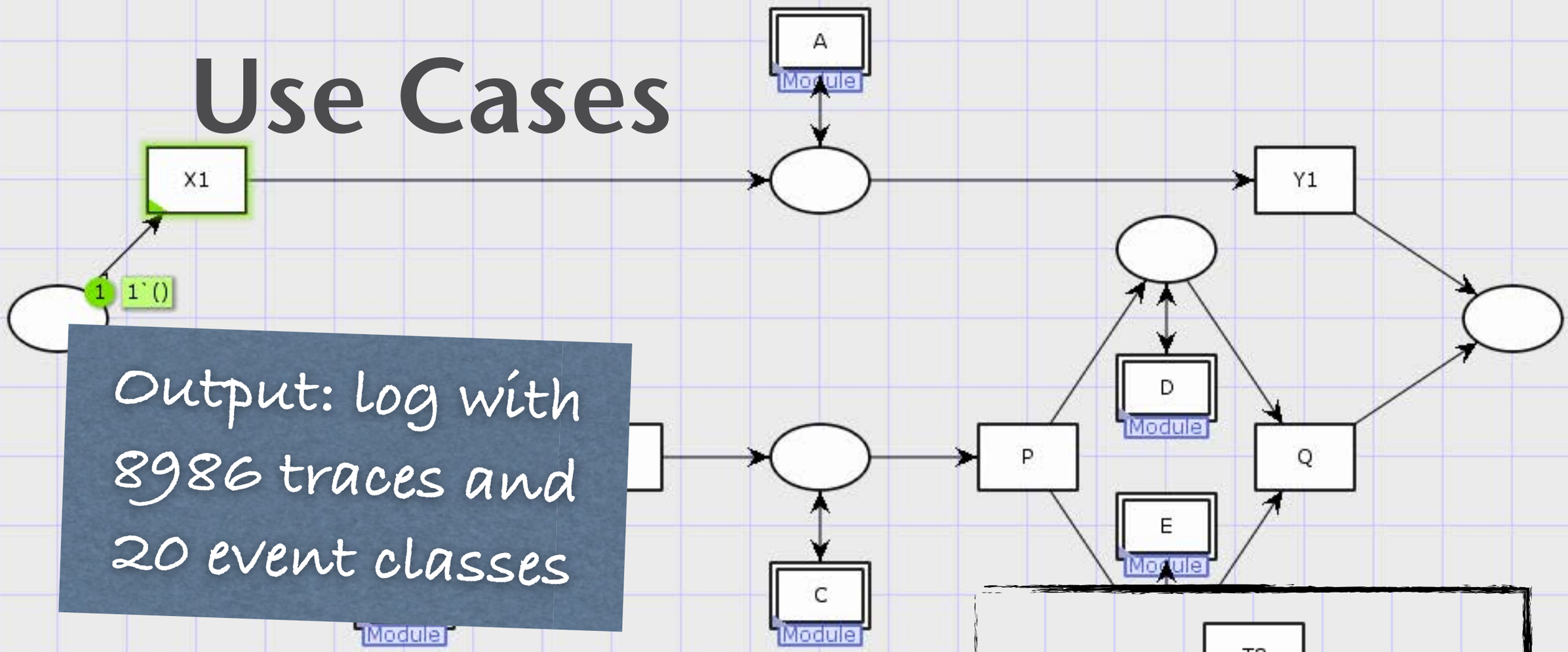
Events of block A and block B can never happen together

not co-existence(A, B)

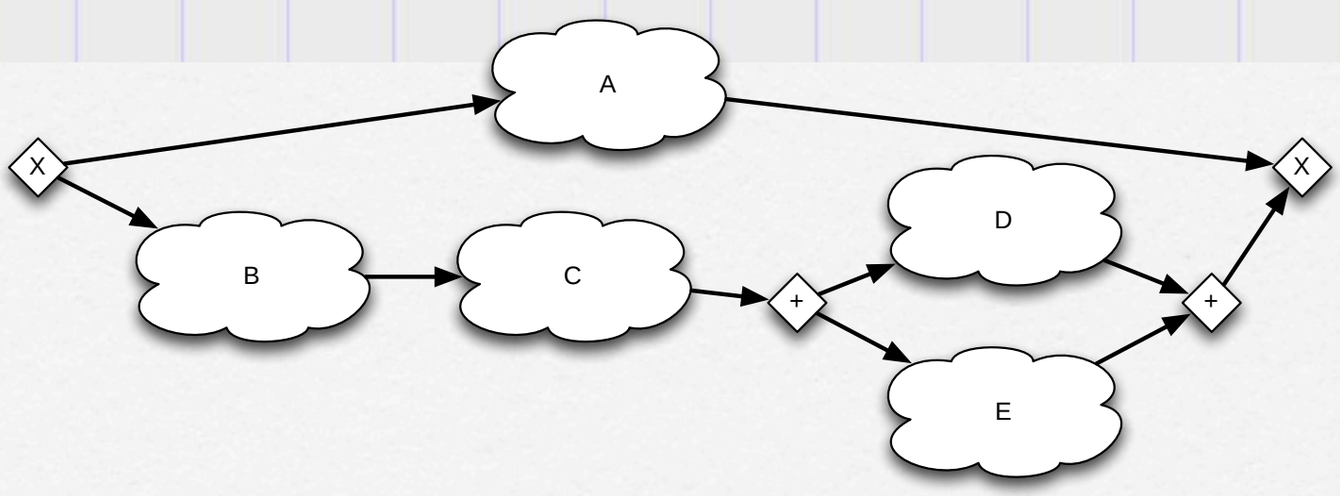
Events of block B can never happen after events of block C

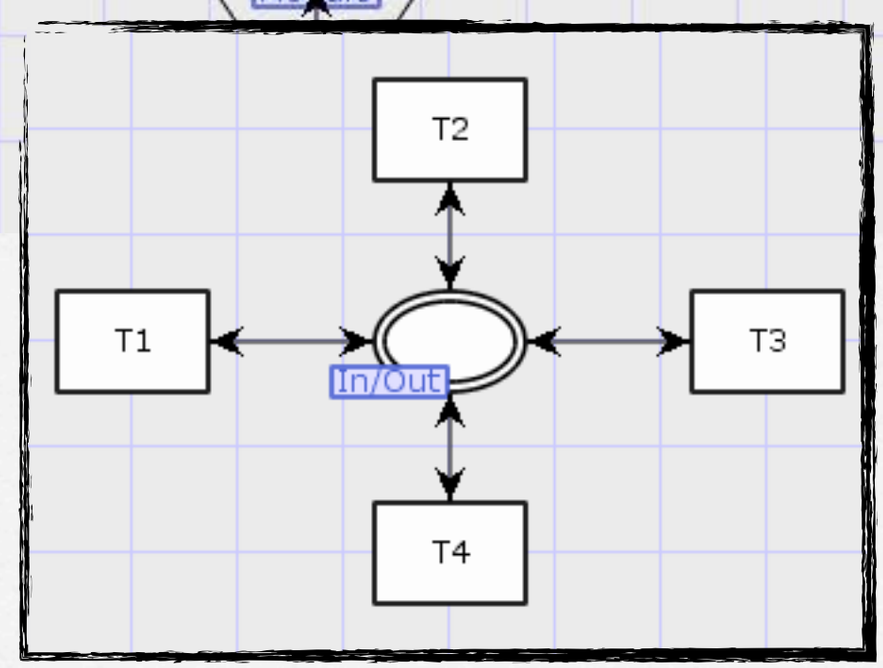
not succession(C, B)

Use Cases



Output: log with 8986 traces and 20 event classes





Use Case

UnconstrainedMiner 1.1 (August 2013)

Log nents/Work/Projects/People/Christian/block structure/log-simulation2.xes Browse...

Output /Users/michael/Downloads/mined_data.txt Browse...

Choice | Existence | Negative | Ordered | Position | Unordered | My Constraints

Constraint	Mine
choice 1 of 2	☑
choice 1 of 3	☑
choice 1 of 4	☐
choice 1 of 5	☐
choice 2 of 3	☑
exclusive choice 1 of 2	☑
exclusive choice 1 of 3	☑
exclusive choice 2 of 3	☑

All | None

Minimal matches (%) 0.0

Constraint | Support

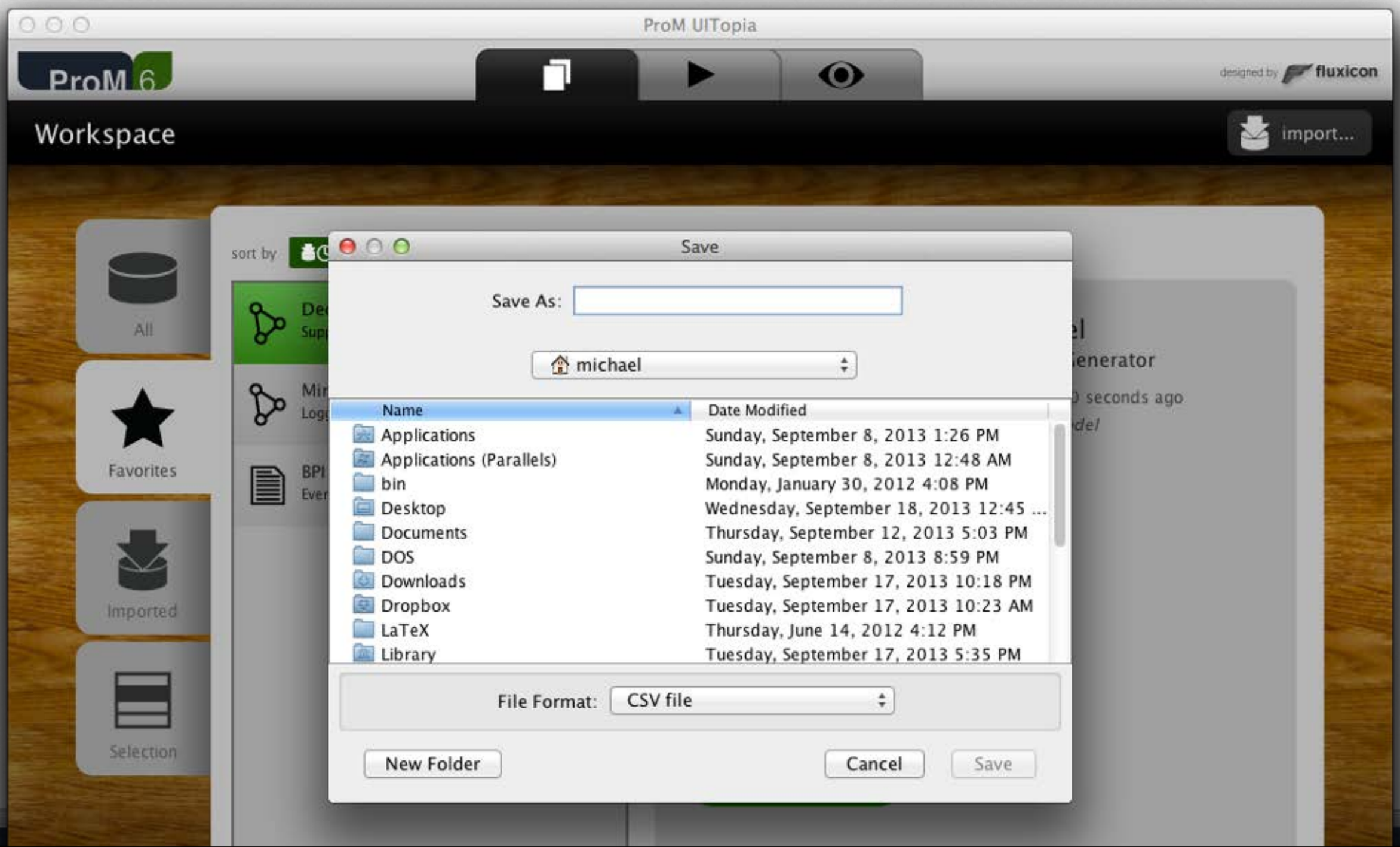
.*[AB].*

	:otherwise	A	B	A
s0:	s0	s1	s1	-
s1:	s1	-1	-1	A

```
  init -> s0;
  s0 -> s1 [label="A"];
  s0 -> s1 [label="B"];
  s0 -> s0 [label=":otherwise"];
  s1 -> s1 [label=":otherwise"];
}
```

☐ In-memory loading ☑ Super-scalar mining 4 computing threads Start

Use Case


ProM UITopia
ProM 6
designed by fluxicon
Workspace
import...
All
Favorites
Imported
Selection
sort by
Save
Save As:
michael
Name
Date Modified
Applications
Sunday, September 8, 2013 1:26 PM
Applications (Parallels)
Sunday, September 8, 2013 12:48 AM
bin
Monday, January 30, 2012 4:08 PM
Desktop
Wednesday, September 18, 2013 12:45 ...
Documents
Thursday, September 12, 2013 5:03 PM
DOS
Sunday, September 8, 2013 8:59 PM
Downloads
Tuesday, September 17, 2013 10:18 PM
Dropbox
Tuesday, September 17, 2013 10:23 AM
LaTeX
Thursday, June 14, 2012 4:12 PM
Library
Tuesday, September 17, 2013 5:35 PM
File Format:
CSV file
New Folder
Cancel
Save

Use Case

10.132
constraint
instances

mined_data.numbers — Edit

View Sheet Tables Reorganize Function Formula List Charts Text Box Colors Fonts

Sheets

Sheet 1

Table 1

constraint	parameters	matches	positive support	negative support	dependent support	parameter1	parameter2
not chain succession	[[4-T2]-[2-T4]]	8986	1313	0	1313	[4-T2]	[2-T4]
not succession	[[4-T2]-[2-T4]]	8986	1313	0	1313	[4-T2]	[2-T4]
not chain succession	[[4-T4]-[2-T4]]	8986	1310	0	1310	[4-T4]	[2-T4]
not succession	[[4-T4]-[2-T4]]	8986	1310	0	1310	[4-T4]	[2-T4]
not chain succession	[[4-T3]-[2-T1]]	8986	1299	0	1299	[4-T3]	[2-T1]
not succession	[[4-T3]-[2-T1]]	8986	1299	0	1299	[4-T3]	[2-T1]
not chain succession	[[5-T4]-[2-T1]]	8986	1297	0	1297	[5-T4]	[2-T1]
not succession	[[5-T4]-[2-T1]]	8986	1297	0	1297	[5-T4]	[2-T1]
not chain succession	[[5-T4]-[3-T4]]	8986	1290	0	1290	[5-T4]	[3-T4]
not chain succession	[[3-T3]-[2-T4]]	8986	1290	0	1290	[3-T3]	[2-T4]
not succession	[[3-T3]-[2-T4]]	8986	1290	0	1290	[3-T3]	[2-T4]
not succession	[[5-T4]-[3-T4]]	8986	1290	0	1290	[5-T4]	[3-T4]
not chain succession	[[5-T4]-[3-T3]]	8986	1289	0	1289	[5-T4]	[3-T3]
not succession	[[5-T4]-[3-T3]]	8986	1289	0	1289	[5-T4]	[3-T3]
not chain succession	[[5-T4]-[2-T4]]	8986	1287	0	1287	[5-T4]	[2-T4]
not succession	[[5-T4]-[2-T4]]	8986	1287	0	1287	[5-T4]	[2-T4]
not chain succession	[[3-T4]-[2-T4]]	8986	1285	0	1285	[3-T4]	[2-T4]
not succession	[[3-T4]-[2-T4]]	8986	1285	0	1285	[3-T4]	[2-T4]
not chain succession	[[4-T2]-[2-T1]]	8986	1283	0	1283	[4-T2]	[2-T1]
not succession	[[4-T2]-[2-T1]]	8986	1283	0	1283	[4-T2]	[2-T1]
not chain succession	[[5-T2]-[2-T4]]	8986	1282	0	1282	[5-T2]	[2-T4]
not succession	[[5-T2]-[2-T4]]	8986	1282	0	1282	[5-T2]	[2-T4]
not chain succession	[[5-T1]-[2-T4]]	8986	1280	0	1280	[5-T1]	[2-T4]
not succession	[[5-T1]-[2-T4]]	8986	1280	0	1280	[5-T1]	[2-T4]
not chain succession	[[4-T3]-[2-T4]]	8986	1279	0	1279	[4-T3]	[2-T4]
not chain succession	[[4-T1]-[2-T4]]	8986	1279	0	1279	[4-T1]	[2-T4]
not succession	[[4-T3]-[2-T4]]	8986	1279	0	1279	[4-T3]	[2-T4]

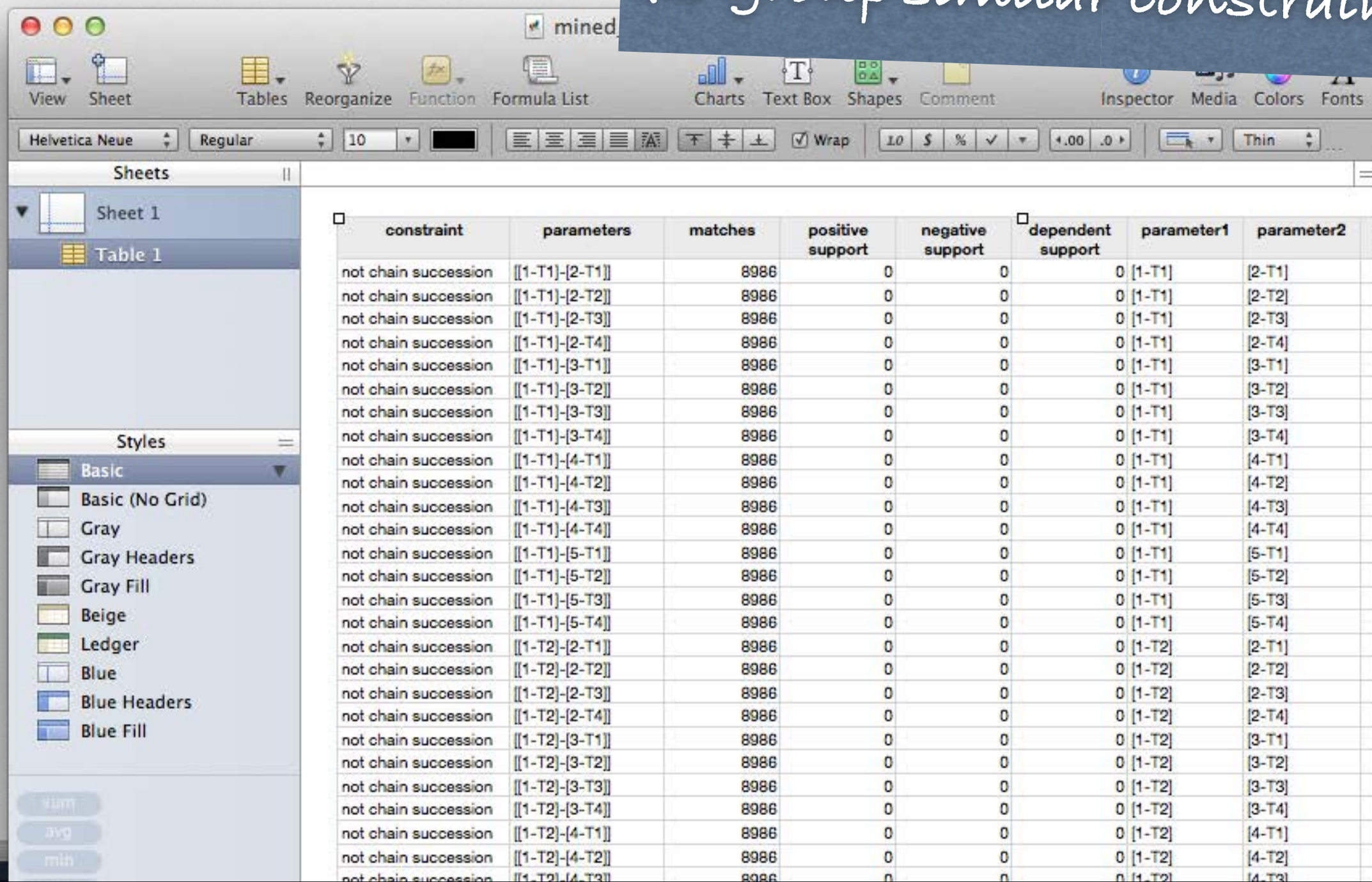
Styles

- Basic
- Basic (No Grid)
- Gray
- Gray Headers
- Gray Fill
- Beige
- Ledger
- Blue
- Blue Headers
- Blue Fill

sum
avg
min

Use Case

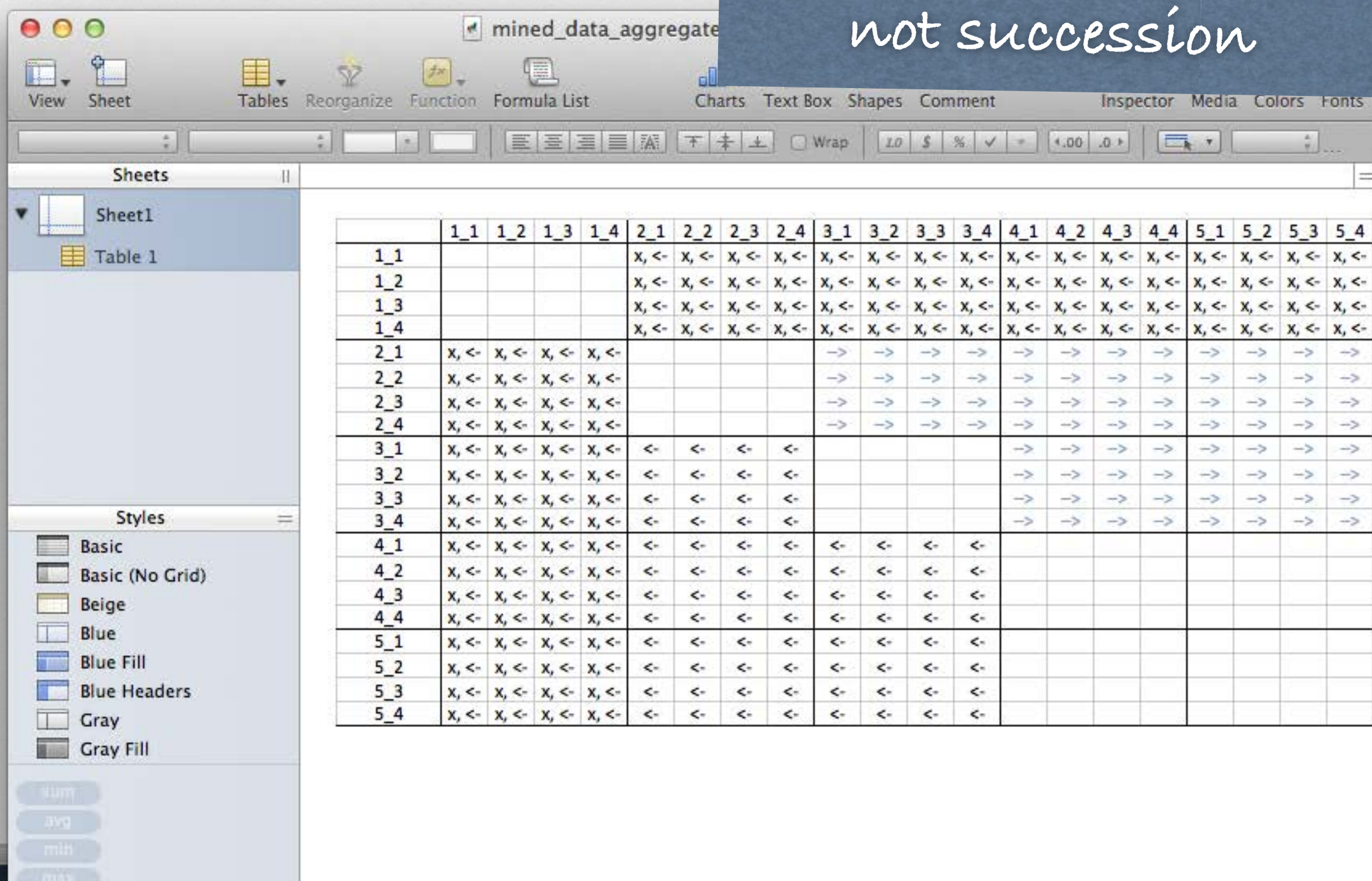
Sort by name and parameters
(to group similar constraints)



constraint	parameters	matches	positive support	negative support	dependent support	parameter1	parameter2
not chain succession	[[1-T1]-[2-T1]]	8986	0	0	0	[1-T1]	[2-T1]
not chain succession	[[1-T1]-[2-T2]]	8986	0	0	0	[1-T1]	[2-T2]
not chain succession	[[1-T1]-[2-T3]]	8986	0	0	0	[1-T1]	[2-T3]
not chain succession	[[1-T1]-[2-T4]]	8986	0	0	0	[1-T1]	[2-T4]
not chain succession	[[1-T1]-[3-T1]]	8986	0	0	0	[1-T1]	[3-T1]
not chain succession	[[1-T1]-[3-T2]]	8986	0	0	0	[1-T1]	[3-T2]
not chain succession	[[1-T1]-[3-T3]]	8986	0	0	0	[1-T1]	[3-T3]
not chain succession	[[1-T1]-[3-T4]]	8986	0	0	0	[1-T1]	[3-T4]
not chain succession	[[1-T1]-[4-T1]]	8986	0	0	0	[1-T1]	[4-T1]
not chain succession	[[1-T1]-[4-T2]]	8986	0	0	0	[1-T1]	[4-T2]
not chain succession	[[1-T1]-[4-T3]]	8986	0	0	0	[1-T1]	[4-T3]
not chain succession	[[1-T1]-[4-T4]]	8986	0	0	0	[1-T1]	[4-T4]
not chain succession	[[1-T1]-[5-T1]]	8986	0	0	0	[1-T1]	[5-T1]
not chain succession	[[1-T1]-[5-T2]]	8986	0	0	0	[1-T1]	[5-T2]
not chain succession	[[1-T1]-[5-T3]]	8986	0	0	0	[1-T1]	[5-T3]
not chain succession	[[1-T1]-[5-T4]]	8986	0	0	0	[1-T1]	[5-T4]
not chain succession	[[1-T2]-[2-T1]]	8986	0	0	0	[1-T2]	[2-T1]
not chain succession	[[1-T2]-[2-T2]]	8986	0	0	0	[1-T2]	[2-T2]
not chain succession	[[1-T2]-[2-T3]]	8986	0	0	0	[1-T2]	[2-T3]
not chain succession	[[1-T2]-[2-T4]]	8986	0	0	0	[1-T2]	[2-T4]
not chain succession	[[1-T2]-[3-T1]]	8986	0	0	0	[1-T2]	[3-T1]
not chain succession	[[1-T2]-[3-T2]]	8986	0	0	0	[1-T2]	[3-T2]
not chain succession	[[1-T2]-[3-T3]]	8986	0	0	0	[1-T2]	[3-T3]
not chain succession	[[1-T2]-[3-T4]]	8986	0	0	0	[1-T2]	[3-T4]
not chain succession	[[1-T2]-[4-T1]]	8986	0	0	0	[1-T2]	[4-T1]
not chain succession	[[1-T2]-[4-T2]]	8986	0	0	0	[1-T2]	[4-T2]
not chain succession	[[1-T2]-[4-T3]]	8986	0	0	0	[1-T2]	[4-T3]

Use Case

Aggregate results for not co-existence and not succession



	1_1	1_2	1_3	1_4	2_1	2_2	2_3	2_4	3_1	3_2	3_3	3_4	4_1	4_2	4_3	4_4	5_1	5_2	5_3	5_4
1_1					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_2					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_3					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_4					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
2_1	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_2	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_3	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_4	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
3_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
4_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								

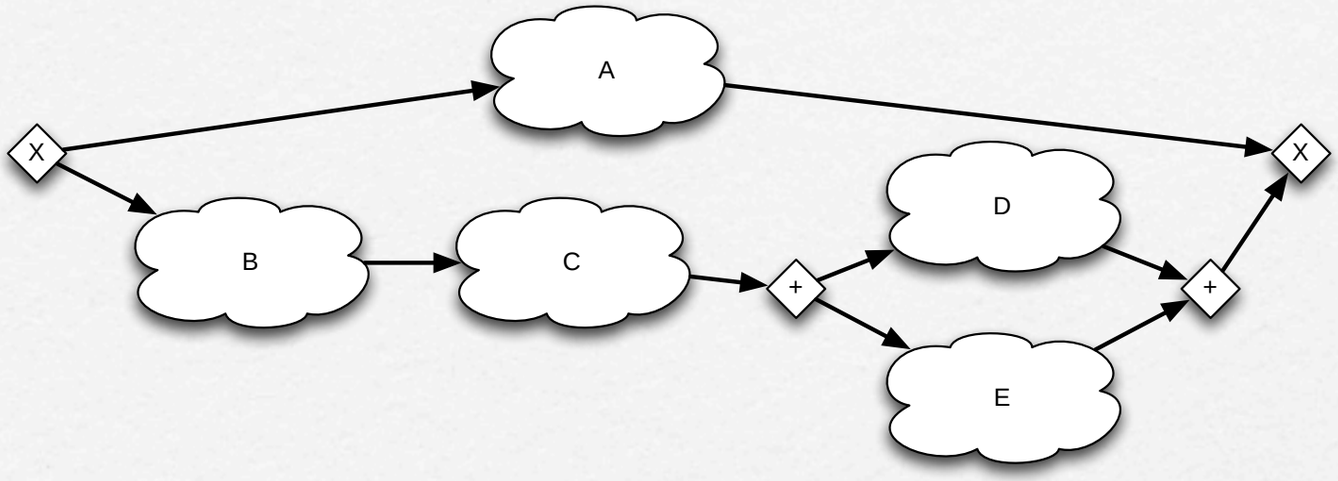
Use Case

We clearly recognize the blocks

	1_1	1_2	1_3	1_4	2_1	2_2	2_3	2_4	3								5_1	5_2	5_3	5_4
1_1					x, <-	x, <-	x, <-	x, <-	x								x, <-	x, <-	x, <-	x, <-
1_2					x, <-	x, <-	x, <-	x, <-	x								x, <-	x, <-	x, <-	x, <-
1_3					x, <-	x, <-	x, <-	x, <-	x								x, <-	x, <-	x, <-	x, <-
1_4					x, <-	x, <-	x, <-	x, <-	<-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
2_1	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_2	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_3	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_4	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
3_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
4_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_2	x, <-	x, <-	x, <-	x, <-								<-								
4_3	x, <-	x, <-	x, <-	x, <-								<-								
4_4	x, <-	x, <-	x, <-	x, <-								<-								
5_1	x, <-	x, <-	x, <-	x, <-								<-								
5_2	x, <-	x, <-	x, <-	x, <-								<-								
5_3	x, <-	x, <-	x, <-	x, <-								<-								
5_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								

...except for parallelism

Use Case



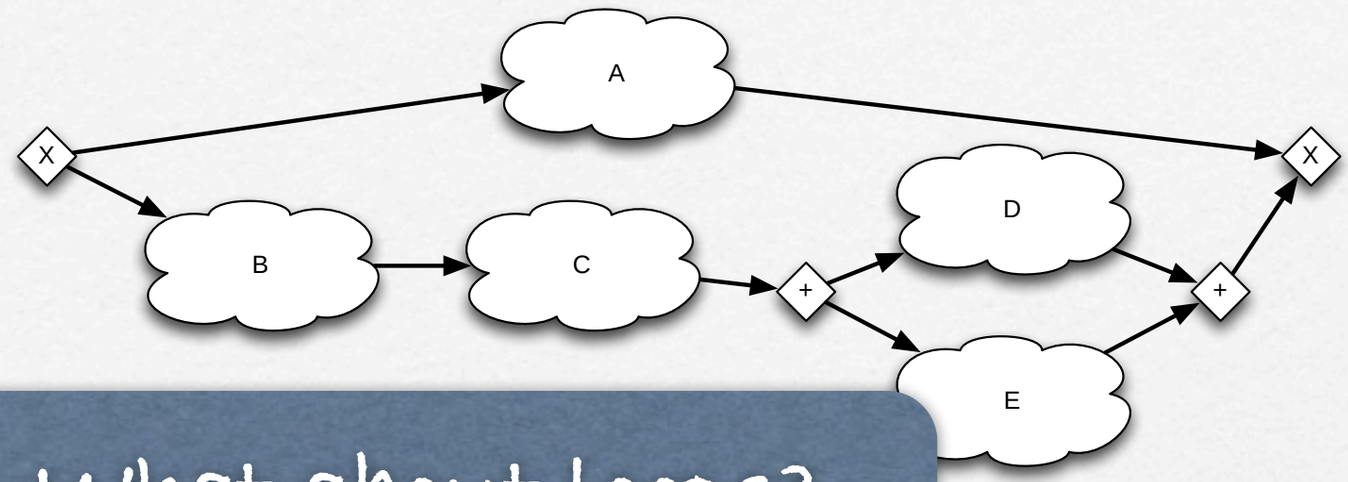
A xor B, C, D, E

	1_1	1_2	1_3	1_4	2_1	2_2	2_3	2_4	3_1	3_2	3_3	3_4	4_1	4_2	4_3	4_4	5_1	5_2	5_3	5_4
1_1					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_2					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_3					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_4					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
2_1	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_2	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_3	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_4	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
3_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
4_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								

B → C, D, E

C → D, E

Use Case



A xor B, C, D, E

What about loops?

	1_1	1_2	1_3	1_4	2_1	2_2	2_3	2_4	3_1	3_2	3_3	3_4	4_1	4_2	4_3	4_4	5_1	5_2	5_3	5_4
1_1					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_2					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_3					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
1_4					x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-	x, <-
2_1	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_2	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->			
2_3	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
2_4	x, <-	x, <-	x, <-	x, <-					->	->	->	->	->	->	->	->	->	->	->	->
3_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
3_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-					->	->	->	->	->	->	->	->
4_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
4_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_1	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_2	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_3	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								
5_4	x, <-	x, <-	x, <-	x, <-	<-	<-	<-	<-	<-	<-	<-	<-								

B → C, D, E

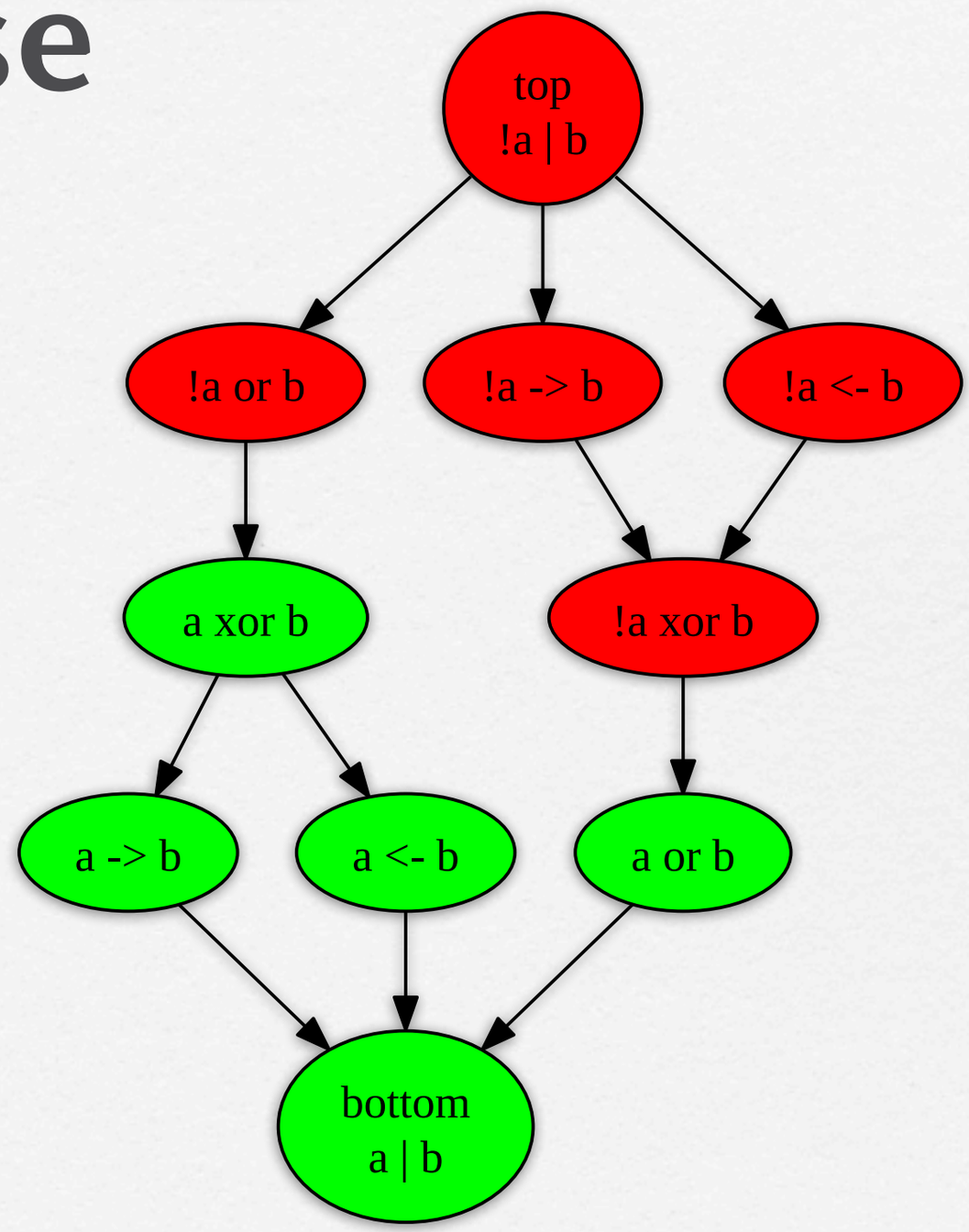
C → D, E

Use Case

Support and
location in the hierarchy
is computed automatically

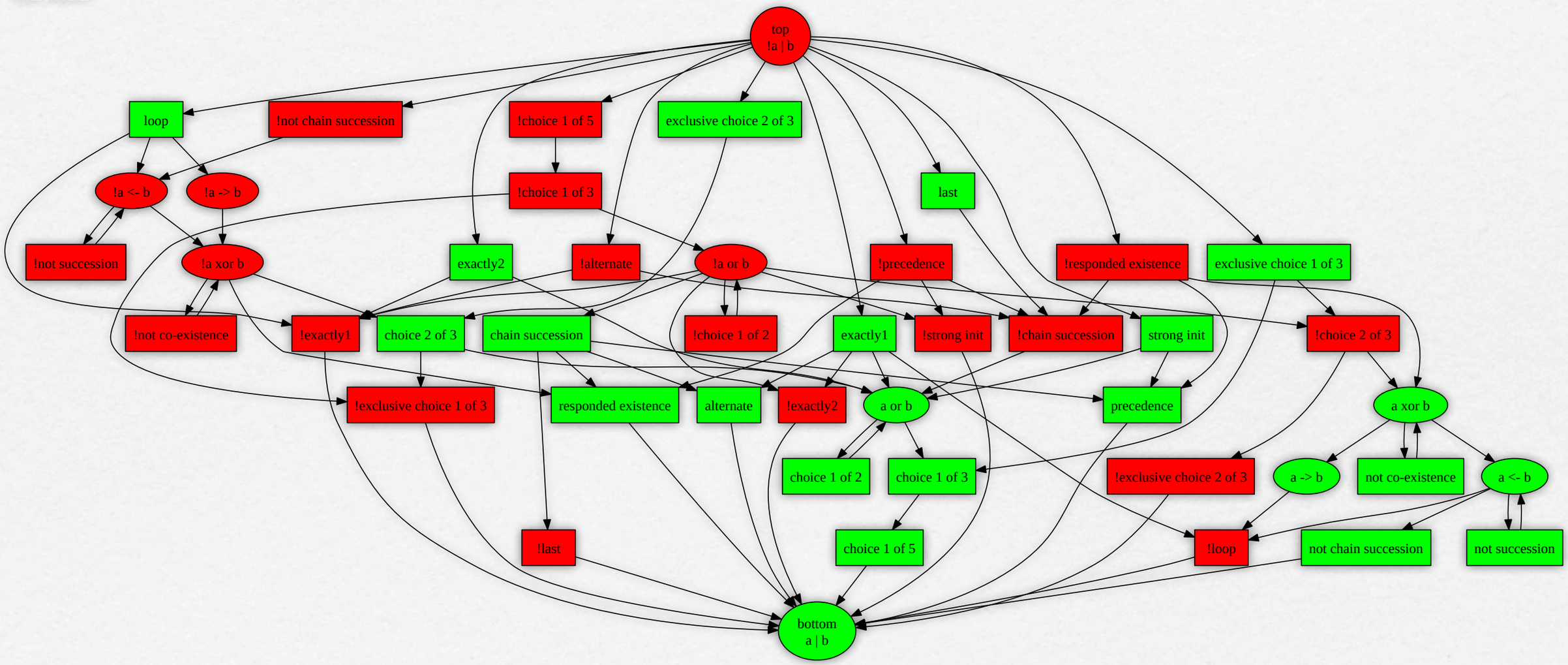
Use Case

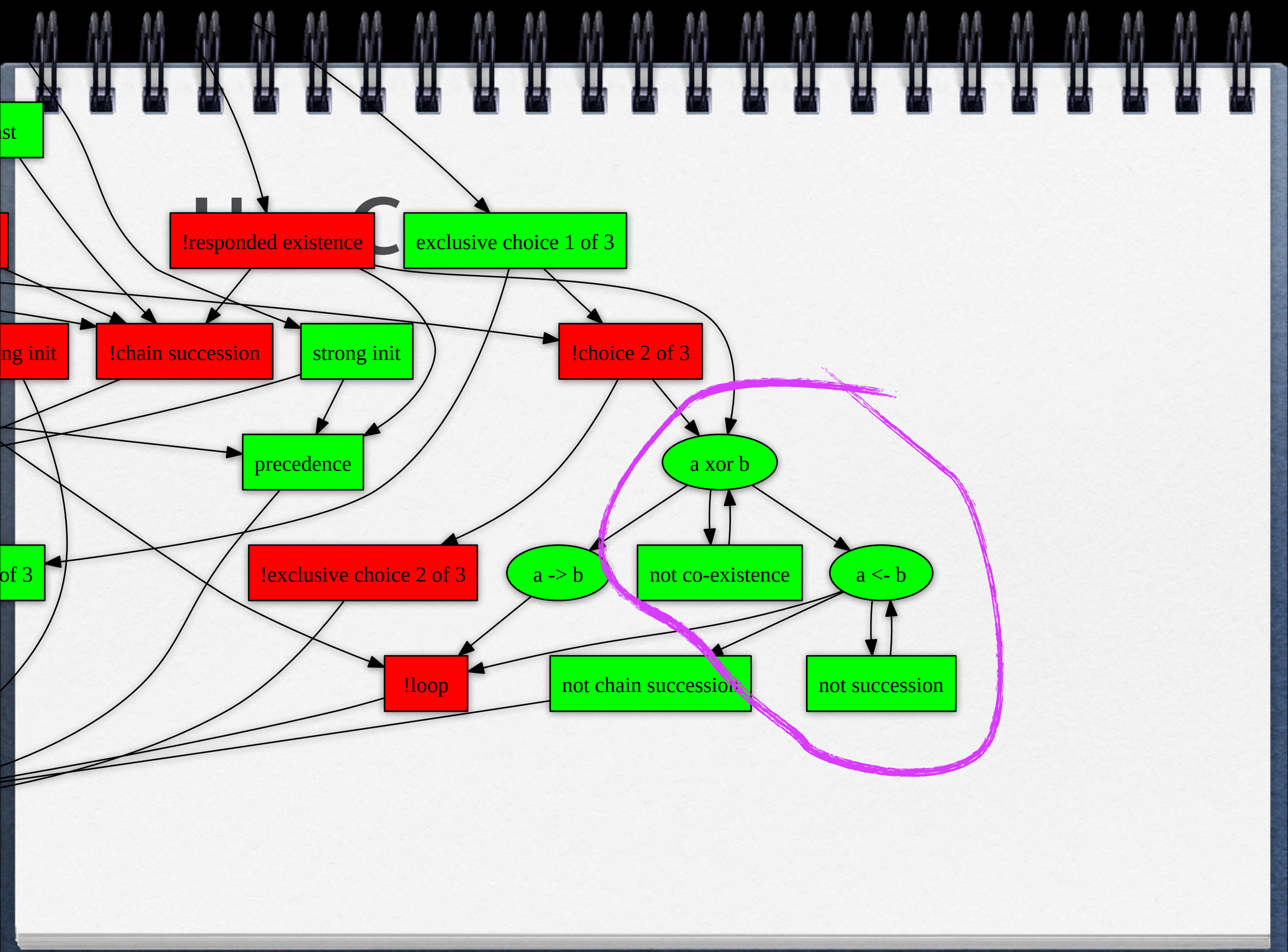
weak relations



Use Case

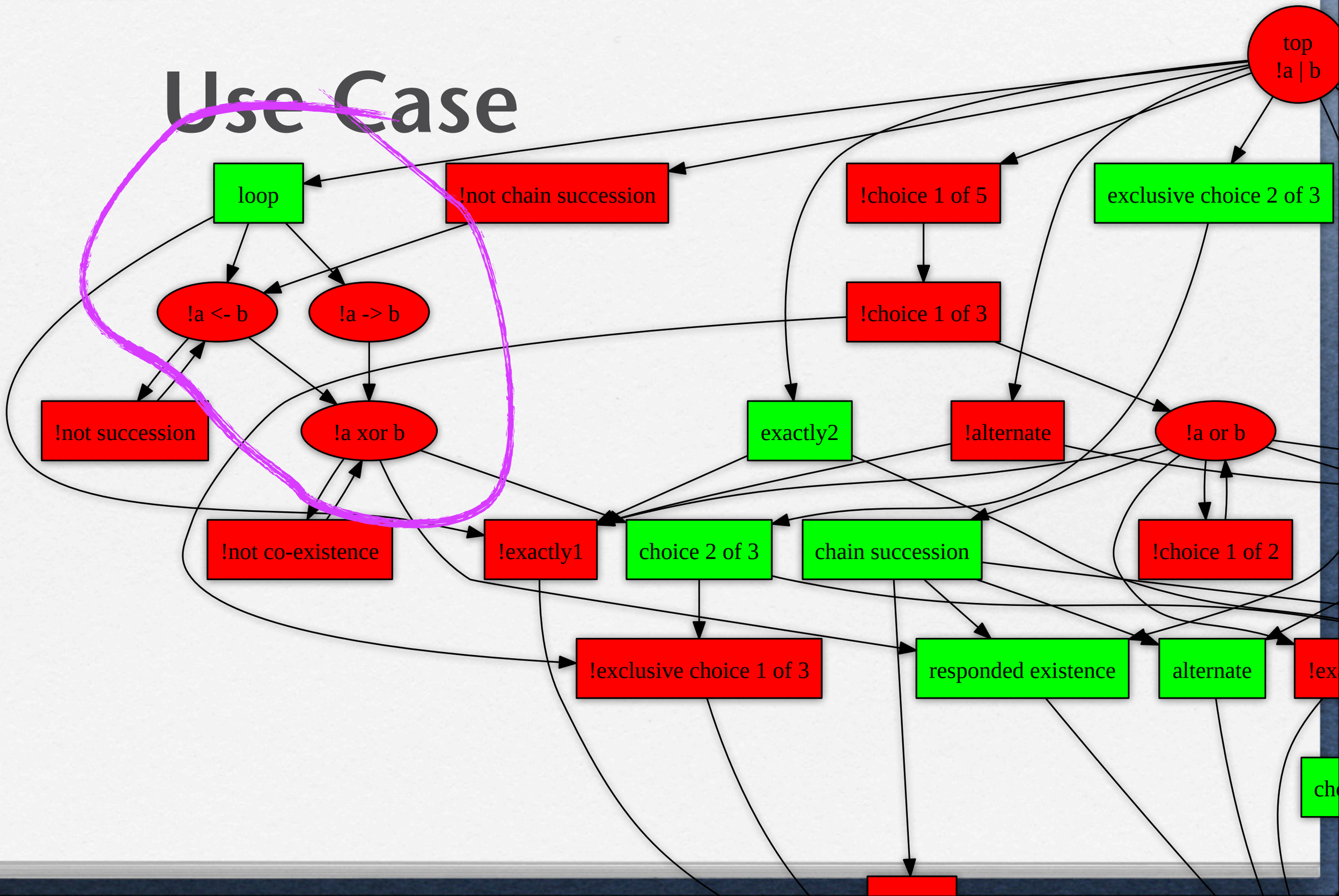
weak relations




st
!responded existence
exclusive choice 1 of 3
ng init
!chain succession
strong init
!choice 2 of 3
precedence
a xor b
of 3
!exclusive choice 2 of 3
a -> b
not co-existence
a <- b
!loop
not chain succession
not succession

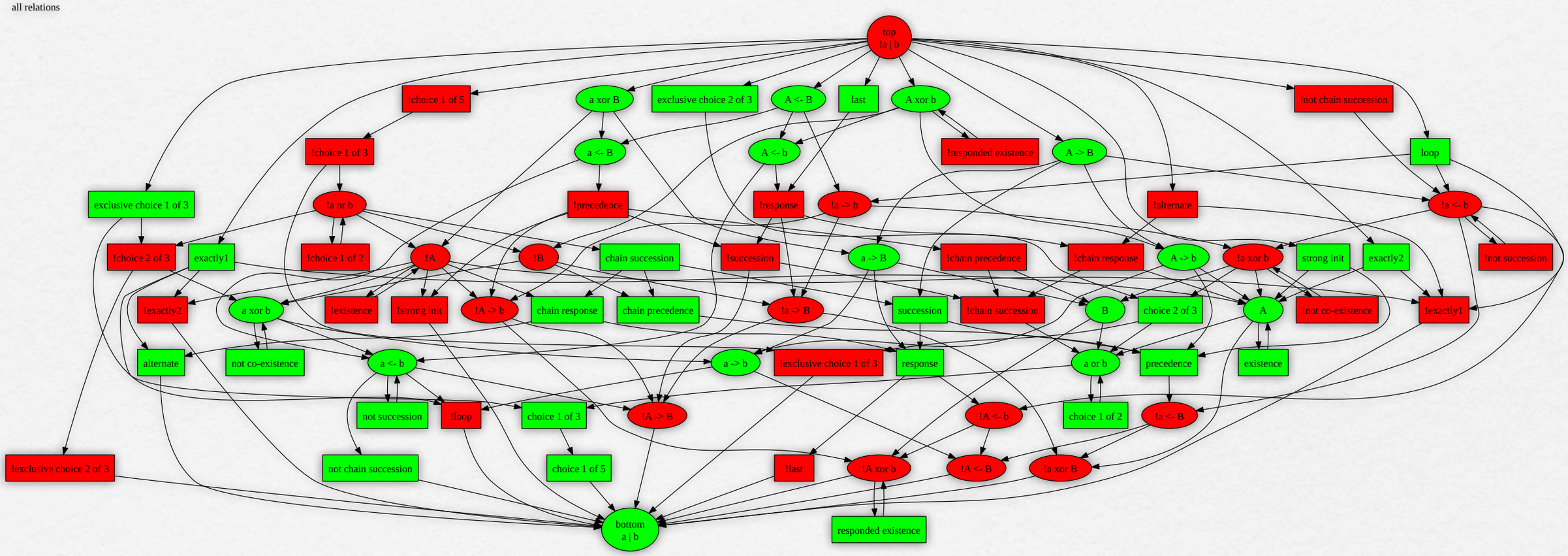
weak relations

Use Case



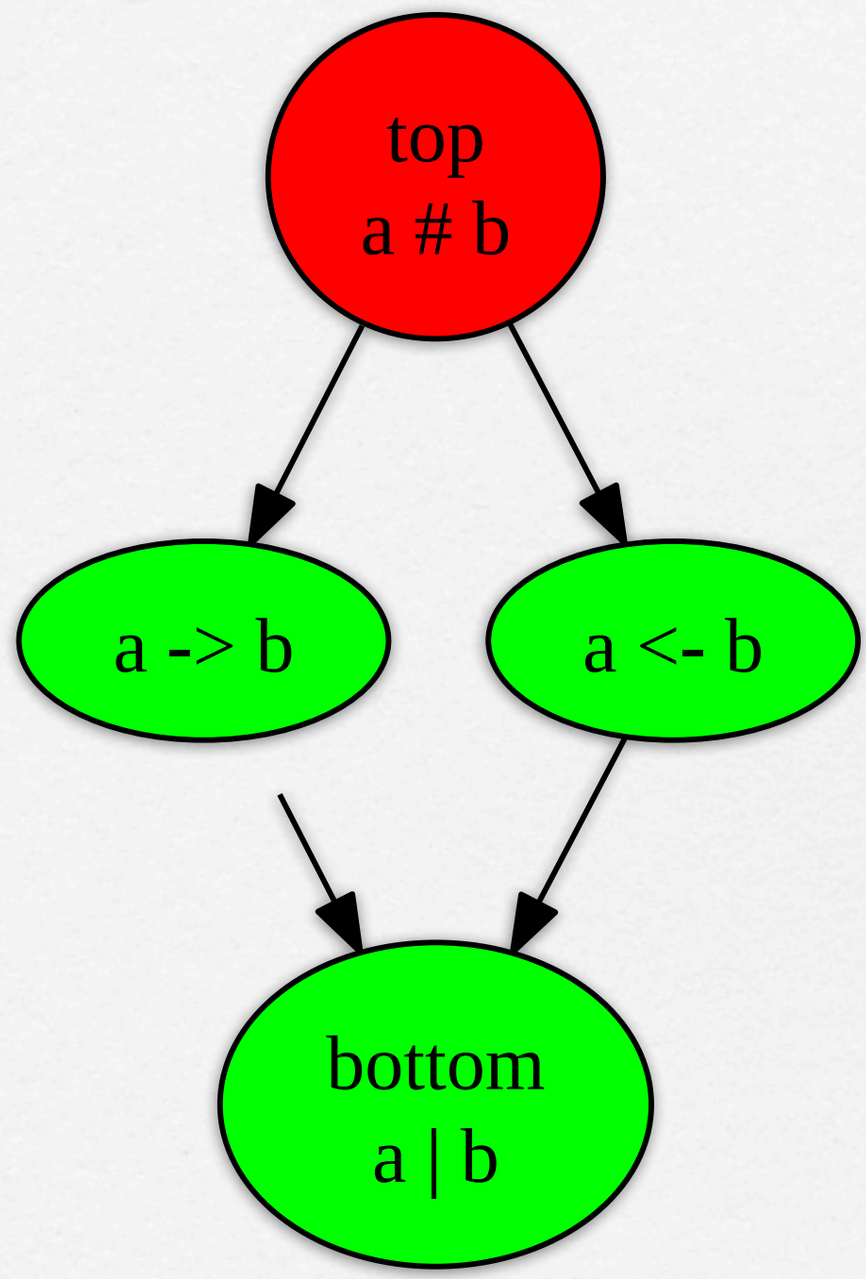
Use Case

all relations



Use Case

alpha relations



Use Cases

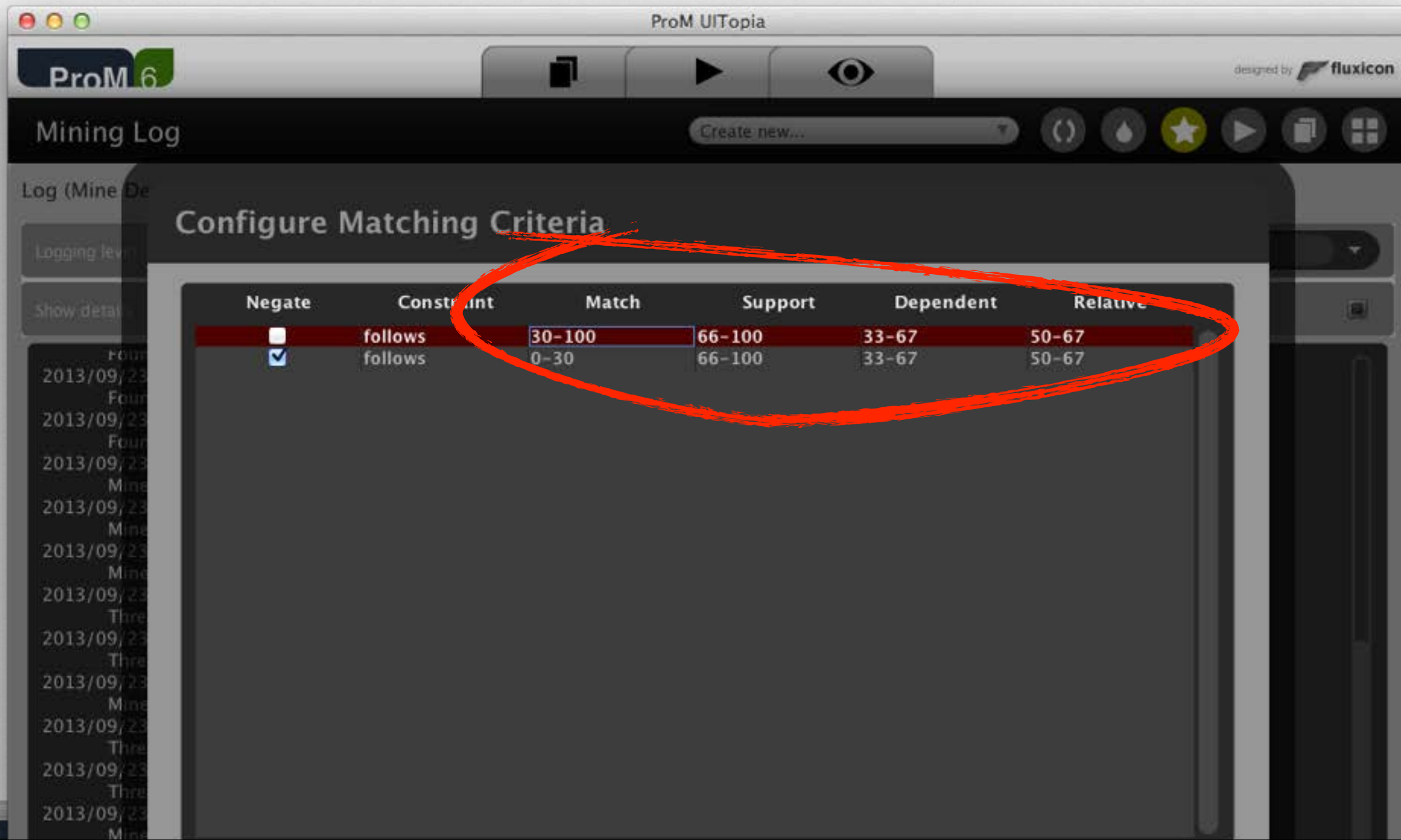

ProM UITopia
ProM 6
designed by fluxicon
Lattice Model
Create new...
A B C D E
A [] [->] [->] [?(<-), ?(#), ?(->), ?(||)] [->]
B [<-] [] [->, ?(||)] [->, ?(||)] [?(<-), ?(#), ?(->), ?(||)]
C [<-] [] [] [->, ?(||)] [?(<-), ?(#), ?(->), ?(||)]
D [?(<-), ?(#), ?(->), ?(||)] [<-] [<-] [] [<-]
E [<-] [?(<-), ?(#), ?(->), ?(||)] [?(<-), ?(#), ?(->), ?(||)] [->] []

Use Cases

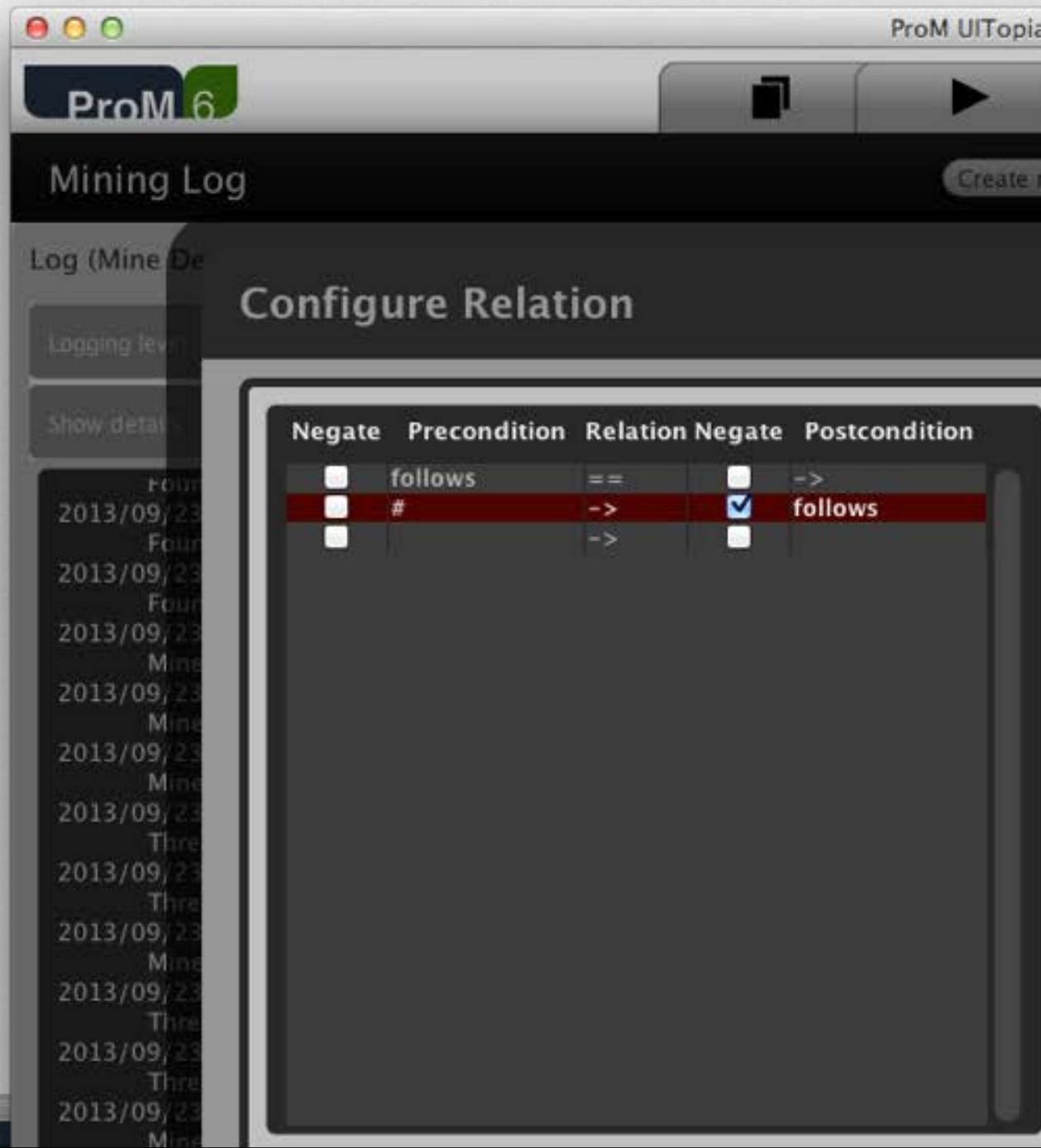
	A	B	C	D	E
A		[->]	[->]	[?(<-), ?(#), ?(->), ?(||)]	[->]
B	[<-]		[||]	[->]	[?(<-), ?(#), ?(->), ?(||)]
C	[<-]	[||]		[->]	[?(<-), ?(#), ?(->), ?(||)]
D	[?(<-), ?(#), ?(->), ?(||)]	[<-]	[<-		[<
E	[<-]	[?(<-), ?(#), ?(->), ?(||)]	[?(?(-		

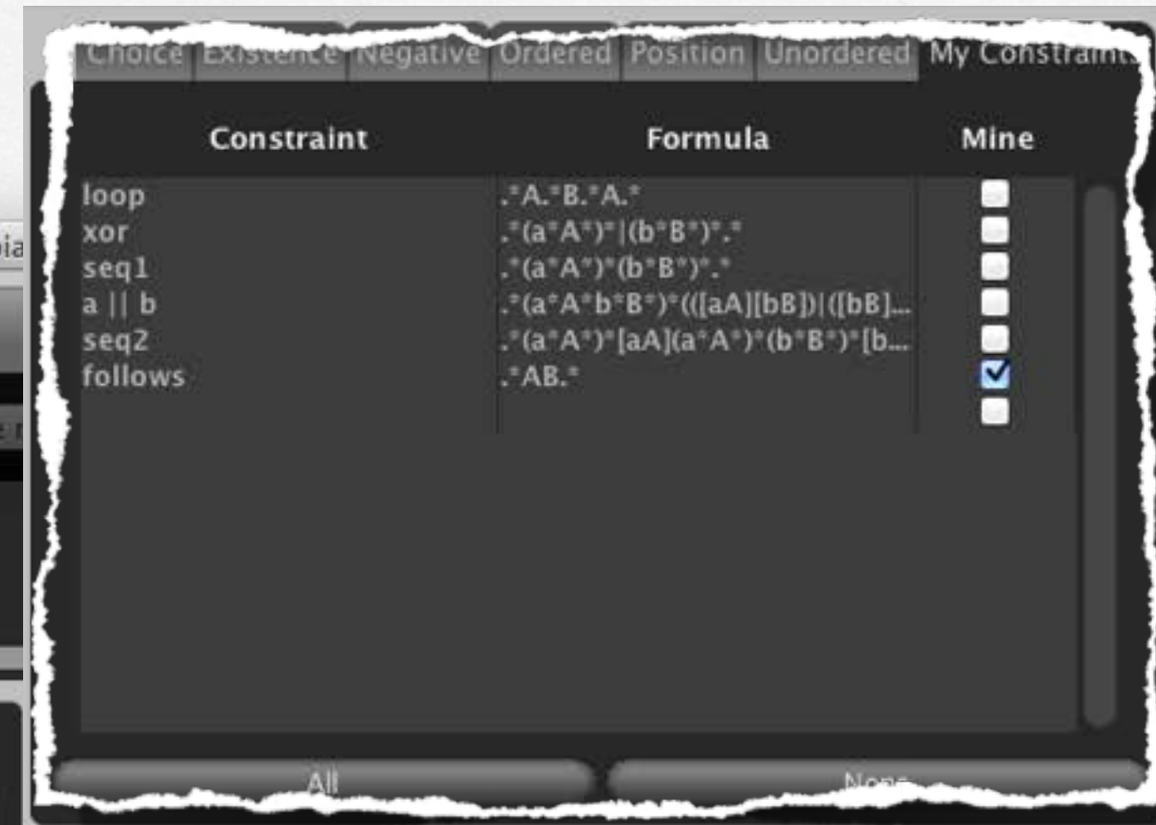
We just implemented the Alpha algorithm!

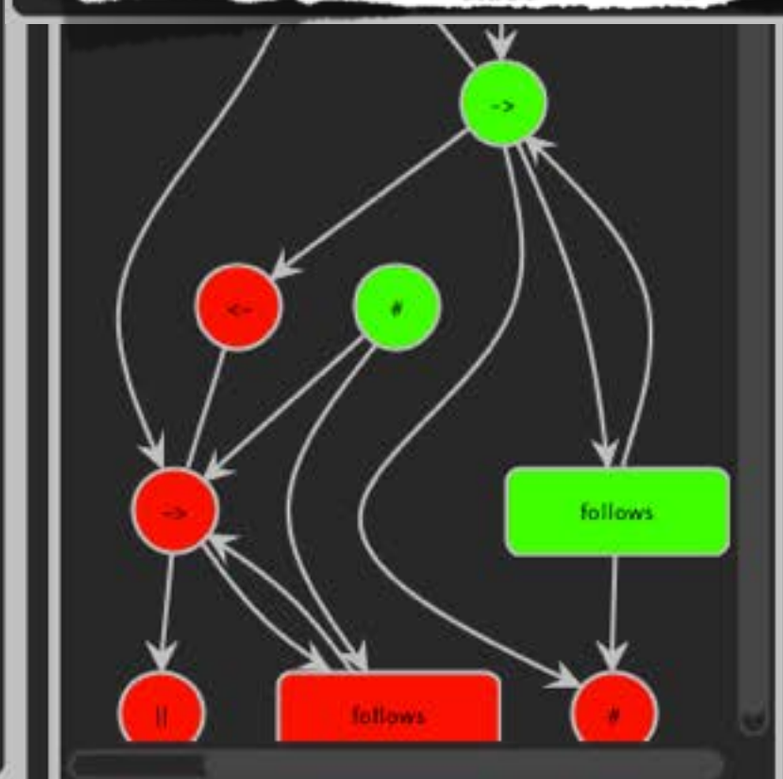
Use Cases


ProM UITopia
ProM 6
designed by fluxicon
Mining Log
Create new...
Configure Matching Criteria
Negate
Constraint
Match
Support
Dependent
Relative
follows
30-100
66-100
33-67
50-67
follows
0-30
66-100
33-67
50-67

Use Cases


ProM UITopia
ProM 6
Mining Log
Log (Mine De
Configure Relation
Negate Precondition Relation Negate Postcondition
follows == -> -> follows
# -> -> follows


Choice Existence Negative Ordered Position Unordered My Constraint
Constraint Formula Mine
loop .*A.*B.*A.*
xor .*(a*A*)*|(b*B*)*.*
seq1 .*(a*A*)*(b*B*)*.*
a || b .*(a*A*b*B*)*(([aA][bB])|([bB]...
seq2 .*(a*A*)*[aA](a*A*)*(b*B*)*[b...
follows .*AB.*
All

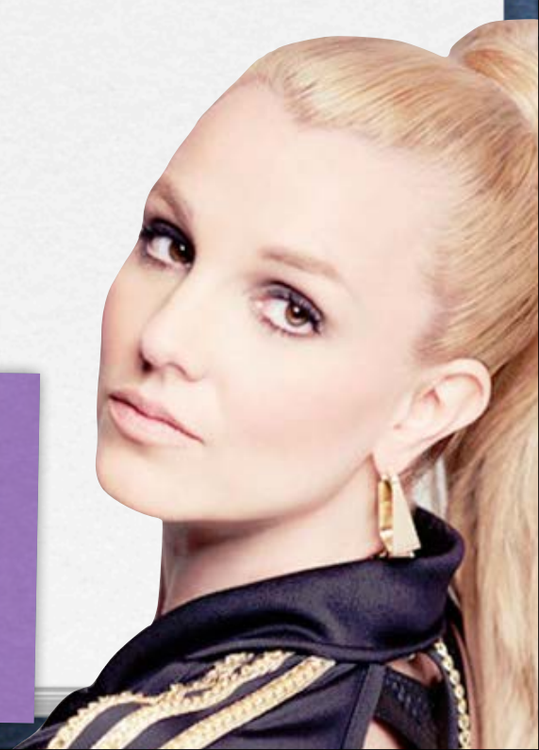

->
<-
#
->
follows
||
follows
#

Conclusion

- Efficient implementation
- Tricks: symmetry, parallelism, superscalarity
- General concept of support

Anything other miners can produce, we can produce

...and we are fully extensible!



Future Work

- Better recognition of modules
- Use of model-checking techniques
  - Directed model-checking for LTL

Conclusion

- Efficient implementation
- Tricks: symmetry, parallelism, superscalarity
- General concept of support

Anything other miners can produce, we can produce

...and we are fully extensible!