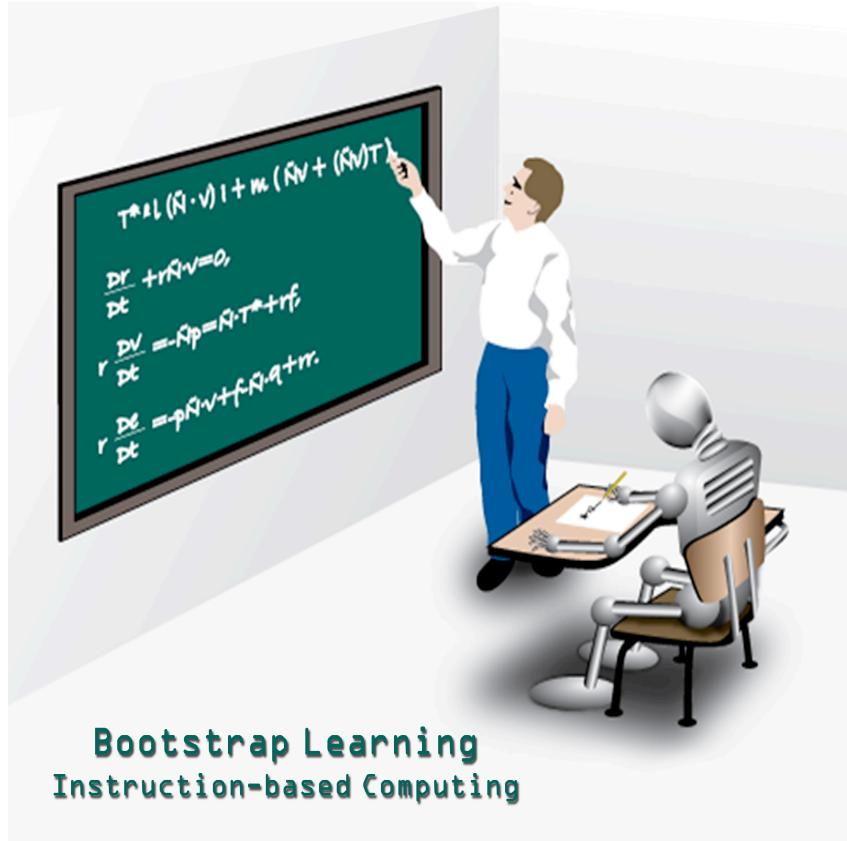


# Planning and Acting in Incomplete Domains

Christopher Weber and Daniel Bryce



# Motivation



- Train Agents cheaply
- Perfect Knowledge Engineering is Costly
- Plan and Act with Incomplete Domain Knowledge
- Focus on Incomplete Actions, but complete set of Propositions

# Anatomy of an Incomplete Domain

- STRIPS: (P,A,I,G)
  - P: Set of propositions
  - A: Incomplete Actions
    - Known: Preconditions, Adds, Deletes
    - Possible: Preconditions, Adds, Deletes
    - Impossible: Preconditions, Adds, Deletes
  - I: Initial State
  - G: Goal

# PARC Printer

```
(:action HtmOverBlack-Move-A4
:parameters ( ?sheet - sheet_t )
:precondition (and (clear) (Available HtmOverBlack-RSRC)
                    (Sheetsize ?sheet A4)
                    (Location ?sheet HtmOverBlack_Entry-EndCap_Exit))
:effect (and (not (Available HtmOverBlack-RSRC))
                (Location ?sheet HtmOverBlack_Exit-Down_TopEntry)
                (not (Location ?sheet HtmOverBlack_Entry-EndCap_Exit))
                (Available HtmOverBlack-RSRC))
:poss-effect (and (not (clear))))
```

# Plan Semantics

- Optimistic:
  - Ignore possible preconditions and deletes.
  - Keep possible adds.
- Pessimistic:
  - Assume possible preconditions and deletes.
  - Ignore possible adds.
- Cautiously Optimistic:
  - Don't assume anything
  - Count models under which plan succeeds

# Synthesizing Plans

- Optimistic/Pessimistic
  - Translate Instance to STRIPS instance
    - Use favorite classical planner
- Cautiously Optimistic
  - Translate to Conformant Planning
    - Use favorite Conformant Planner (count models)
  - Default Planner
    - Count models
    - Count failure diagnoses

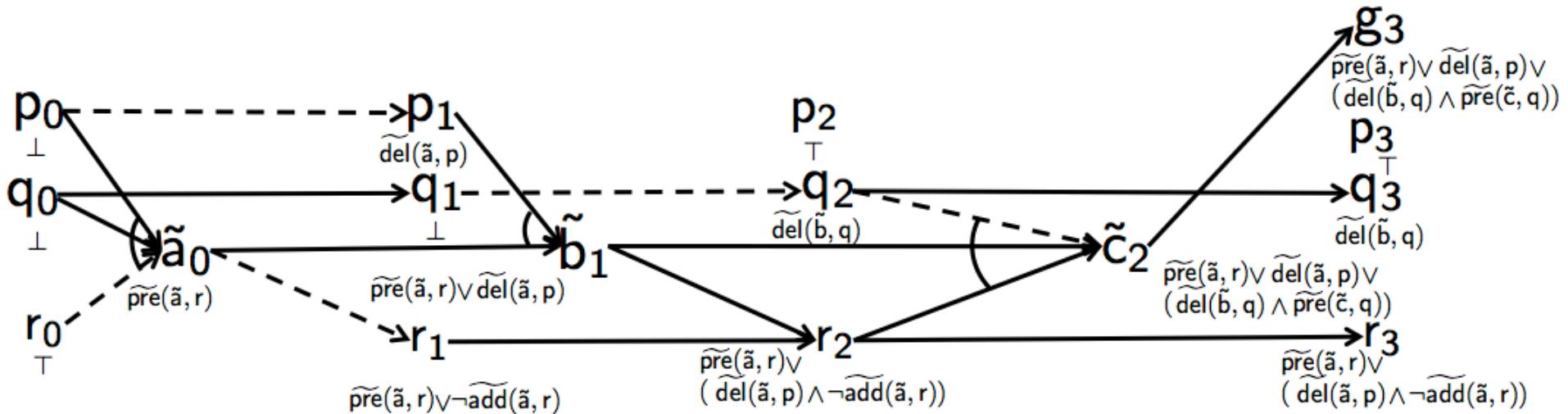
# Conformant Planning

- Action Incompleteness → State Incompleteness
  - New propositions (e.g.,  $\text{pre}(a,p)$ ,  $\text{add}(b, q)$ , ...)
  - Initial Belief State (e.g.,  $\text{unknown}(\text{pre}(a,p))$ )
  - Condition each effect on incompleteness
    - $\neg \text{pre}(a,p), r, s \Rightarrow g$
    - $\text{pre}(a,p), p, r, s \Rightarrow g$
    - $\text{add}(b,q), m, n \Rightarrow q$
- Must be Conformant Probabilistic to count (weighted) models
  - Conformant Non-deterministic requires strong plans

# DeFault

- Extend [Garland and Lesh, AAAI02] and [Robertson and Bryce, HDIP09] from single fault to multi-fault.
- Foundations in Model-Based Diagnosis [Reiter, AIJ87] and Assumption-Based Truth Maintenance Systems [de Kleer AIJ86].

# Plan Failure Explanations



Propositions = { $p, q, r, g$ }

Incomplete Actions = { $a, b, c$ }

Initial State = { $p, q$ }

Goal = { $g$ }

Incomplete Features = { $\text{pre}(a, r), \text{add}(a, r), \text{del}(a, p), \text{del}(b, q), \text{pre}(c, q)$ }

Complete Features:  $\text{pre}(a) = \{p, q\}$ ;

$\text{pre}(b) = \{r\}, \text{add}(b) = \{r\}$ ;

$\text{pre}(c) = \{r\}, \text{add}(c) = \{g\}$ .

State Sequence: ( $s_0 = \{p, q\}, s_1 = \{p, q, r\}, s_2 = \{q, r\}, s_3 = \{q, r, g\}$ )

# Propagating Failure Explanations

$$d(\tilde{a}_t) = d(\tilde{a}_{t-1}) \vee \bigvee_{p \in \text{pre}(\tilde{a})} d(p_t) \vee \bigvee_{p \in \widetilde{\text{pre}}(\tilde{a}_t)} (d(p_t) \wedge \widetilde{\text{pre}}(\tilde{a}_t, p))$$

$$d(p_{t+1}) = \begin{cases} d(p_t) \wedge d(\tilde{a}_t) & : p \in \text{add}(\tilde{a}_t) \\ d(p_t) \wedge (d(\tilde{a}_t) \vee \neg \widetilde{\text{add}}(\tilde{a}_t, p)) & : p \in \widetilde{\text{add}}(\tilde{a}_t) \\ \top & : p \in \text{del}(\tilde{a}_t) \\ d(p_t) \vee \widetilde{\text{del}}(\tilde{a}_t, p) & : p \in \widetilde{\text{del}}(\tilde{a}_t) \\ d(p_t) & : \text{otherwise} \end{cases}$$

# Counting Models and Diagnoses

$$pre(a, r) \vee del(a, p) \vee (del(b, q) \wedge pre(c, q))$$

- Propositional Models: 26

- PI1: 2

- PI2: 1

$$pre(a, r) \vee del(a, p)$$

- Propositional Models: 24

- PI1: 2

- PI2: 0

# Models (26834 Incomplete Features)

180121028890584300290729584160004996156207796772280973660887554626675655849972216741038196107977898899390666530957343736531263779734368545984171433240849822345541333057  
1585497779999218421429059419542882655752492779515306855663330805388163099539410241622433360022506846766911679054473633695145123746602008388289773699277249804741053079  
406729519009202260767319751345138506273192307747205765935525888723574447391010728956635268161973920159054150979931237137747806939501804423313191349502790298233100487846  
727317831551843556606339532709560801118116320033501567779190110126120050083261941009249961412506563934025947726468867854565155447810300119857708327343535140002892683356  
700939628951695123825120951519116715890684004527924149096252318773937686926813576564714144446170239858662275668281439533697835739430686928001092866564120831906312890  
80324348788857397098948469409531356600854129294839142718065435313446792917193660704439729105290313075436196159419451321896827858144157335460908675197486103326857522145  
319268435485247656935784622087954260027470101217836947452017269971557419767840085303643799093678336879602268636768422026634464028942402890705667052413432764796610926444  
9573459082392918142316237760463189322196728074126550789635768460627243249261566326153709271565806140497615610078835807174202280759572336707134573282230378274105684115  
104071073680323487415980789720011846430075906433188395929906722118108625448609735599283218774418966428618807184061829183004785981348576332640502620580665771640219017  
13903738131205936095395790896441873692044498979772595511293967681719230770906788569742017962874219578424975107960877365279973152897210072603447128344273139451642706223856  
621819453701899122898781834456131432384296723931797251093277871250341211543681057996074008880535083997678433580000961327996899213691863452030572693116824311219241265938  
49978699646668143150779810843823816805797386207872092740051117260379708600518888313892951316370901966134314697148483535303141718010237802921114360931426590335502578320  
424993529170377714711177931342081712709392909360069627078532325465478391102221936893377866583322300799286918743636526824375674220262685681543246502255082378491658977978  
1836308840506471381163898903082035198047983678915376951894421775781293124906671620606358743201384879516388487634749402462718632505268496343248695986950856853815067448  
247190951010804486468916419632597857093313808317051123304355039182927441022331347809387944449020985481411268278791465769341284496381531248720008577253850017880024707703  
635275380227764067575366606267862054001850668741928937397909863281360261102337569415320943585087974753219318900830686002208827886499501832616024603421021717584465471  
7785705423963376186517694722012551083354353763887762119323721583944071130285941133828845137061152780075014914619810525198373430168174341546491548392500556574913564  
692701337552173723387384630469770636699051189332151582667047444357553023693702207264428372780805743531103859120657659859141985508943024842392324055311557926212378610893  
066827774244669429790793566232522315647014131450859514554218398608169941456105912626677883166211991582766320806120124438523298410401392047574310328866093688327357332787  
453971009745432112490453116203681727114038522430715793894477233307216585439974722443172581358757368840374854900550086588740840443658043882270674720699892221717007568  
77313718525789643662056074999411497305027273365768939251508280848946144658110909814016511015947952086170904553464124257915120802019328074123422016037935941988342776111839  
63609197024011181124154533216869275517425574406778897398204059906528379226433476021200435560346410095954755924052775734062384558337897232416491942121019393150355975850  
6323203408794621226940348120311320558921111434802868064145699178545828414589500829185632922411036445087173393779226823565756411883914471299740481125590415054984821  
921075757664055146731726792654436478048462769065716480980434745417841203696767582943079443157482373441704428463737534093948477854921591548463404146759782302529647567380  
17594522205037234621362748173108587746505189388457645103291480825575023927169919177476518260862145474064511833413609034601688248697084060276200659807524568435605  
751975611321387859672387165527726216385243776548757969243826856472718346355839397355673615377641579506060699324050041550624512339021742424930802151421670466343224406006  
843793079373236527471361501659921581203112687337985514453894181806198430395422877548617538724604209722454578622182257207664921968526119753491353849686911163575791423  
26795389657301866293554591097358221614366629864360206292649534780190533809720376475520346235890375255477621205698239051787065684738602641038230926103859206511937389913  
959922408024270332985966968431098590835731043597934657755131012086285251583286716943343892521812241469951417024656845428513442984386716275098263804376660218510754654132  
33439676583937979155199250203126127104865908271778535566690041283918604184096381572096964248501668231004486356429090515181292986605773908302718528660953032181200927901  
85199306418239432960908466635471686843806324313656462292702973682584821038042295127094087050122309238632361675229359792052358942190955261963575640225364565476580  
197650605159611473110510463743889585737367488931125436395635660459773279448617048116152672978104328313109352081819454788973363935939815721852345164335119305965803395  
88718346563013671017308220588316354166650705564755450409165710785212850247600499246770539037453013472481591931614813220543221730202219347270396943837358529619706915137  
33260910026136467747522074214140791125587774658638759842274128462130043084358248493613034334102479980311053695136438305022283470877469229633224679620262938570352  
09462660138410658144184979121656717219879133672627354962950925360163658576790927812372414222882875867068468063193061192521511917072100724332786442766029274944494846638394  
3585436952553019648135120608633053731231784878317418486040958677480859472828994112737488048608546286699788537960744775774480685882075226409156847931780427  
816978689579284064330126303839032564326695112592716871700432843881090277501446455363619939999760094149348234991993257151256548447983506463641771353362075718144112471209  
5631660212784703811823549035543327401487990082932310074199291433365629060119584995058985460189967908030219317719198734637210004363075222568679579678093625499276481700  
39301929655367471098530014768954681522036557946455671913984339981705343121571842163929905268473954715613682952433727953247887383645533276523126146178860961075389058665  
99904172883248069480243750905547164073756212523020719003889933413396595992739914909572902645310455975494805505411030018615691761826383645201766485604070725350029  
370067528254480662492815803952518402734861419163592214954364351675198597323430397578874981223593094505460061082706956893523150077846176665886726513725645913258332810756  
9938879270059828597199526075938243416643734400749385330069886133074001096191833232623163494105248976312018262187664662093811740153469483213634652676594448251787417  
343752714612797349869297611062987211724674574665565410616230478029804075530744039672091735725105863793282092091273654435927767209012191809138252410313997015307050848  
633560873628746974373315619881537882343922145597117789700038712247074620199522149345790778709236779608883392119669973934898718468229553130545719213680911512023860132809  
08697869793725167488332291792209614730037304342574165212558127639144010047889431916660612027111262527158932344758846012664788702988174609331280358573308948  
7561923866972901534400916153140729099033396071974215497398103694983043898559630691261698212533473349246975981036106277763632541961974225234569950532502698883205230570  
60608579521507403545003322948543717051384541910751384541907513845393004217966499543035556387370890121448337335669454969183042415605302168429539176864074445569952546440621767269  
04183068764721969831913537524162108321293830557294762396

# Heuristics

- Optimistic/Pessimistic
  - FF Heuristic
- Cautiously Optimistic
  - Propagate PFE's, ignore deletes

$$d(\tilde{a}_{t+k}) = \bigvee_{p \in \text{pre}(\tilde{a})} d(p_{t+k}) \vee \bigvee_{p \in \widetilde{\text{pre}}(\tilde{a})} (d(p_{t+k}) \wedge \widetilde{\text{pre}}(\tilde{a}, p))$$

$$d(p_{t+k+1}) = \begin{cases} d(a_{t+k}(p)) & : p \in \text{add}(a_{t+k}(p)) \\ d(a_{t+k}(p)) \vee \neg \widetilde{\text{add}}(a_{t+k}(p), p) & : p \in \widetilde{\text{add}}(a_{t+k}(p)) \end{cases}$$

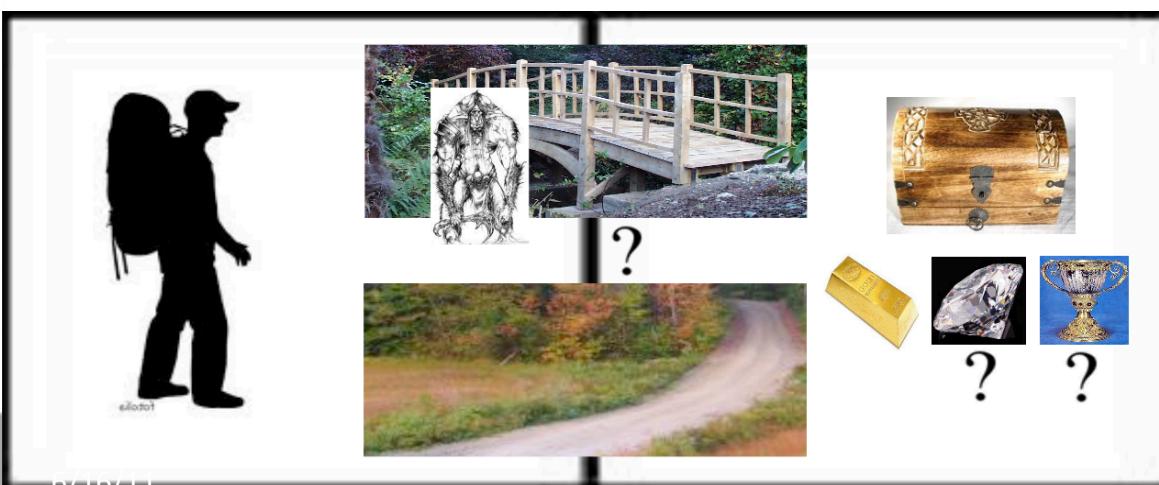
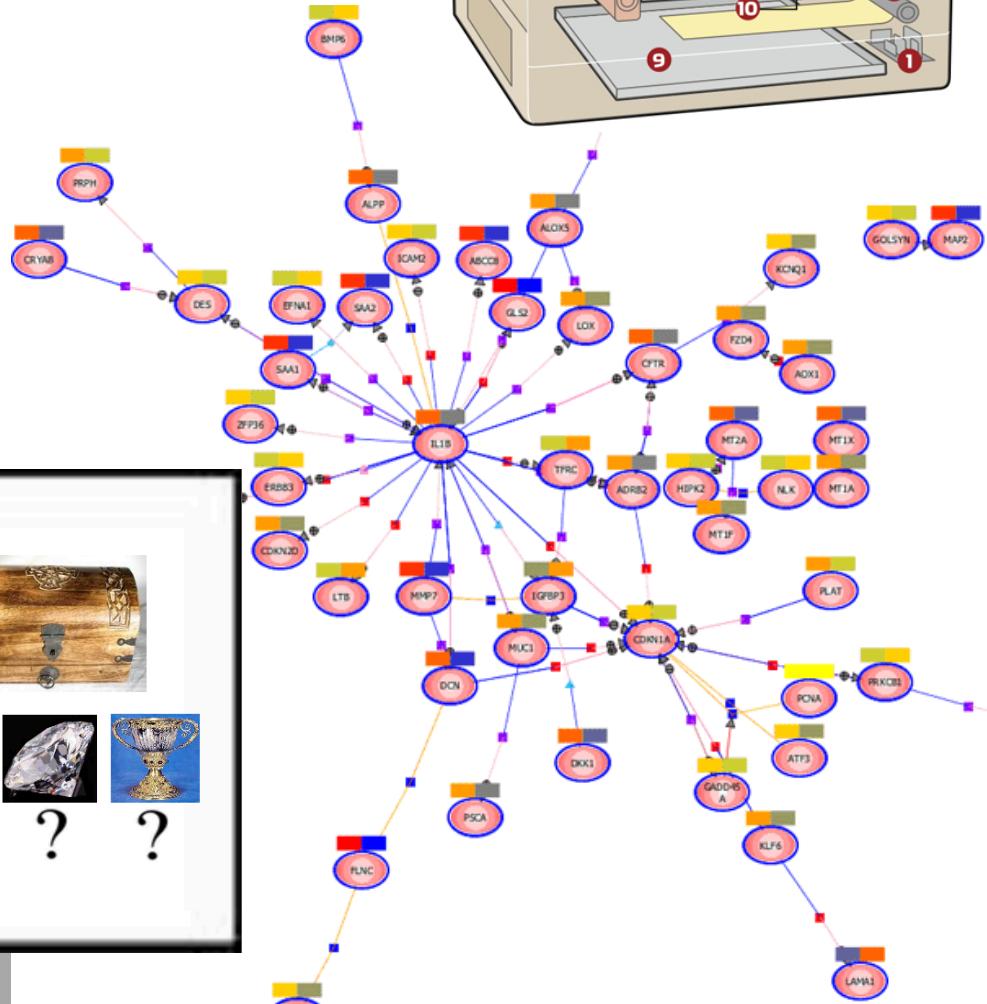
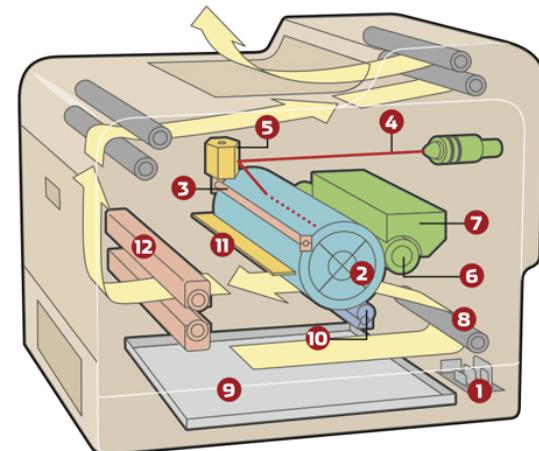
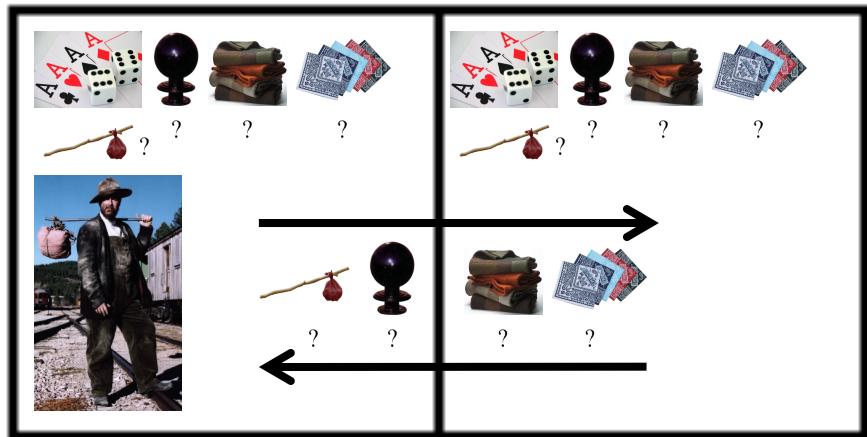
# Heuristics Cont'd

- Select Supporting action for each proposition
  - $h^M$  Fewest Models of effect PFE
  - $h^{PI}$  Fewest Prime Implicants (Diagnoses) of effect PFE
- Extract Relaxed Plan using chosen actions

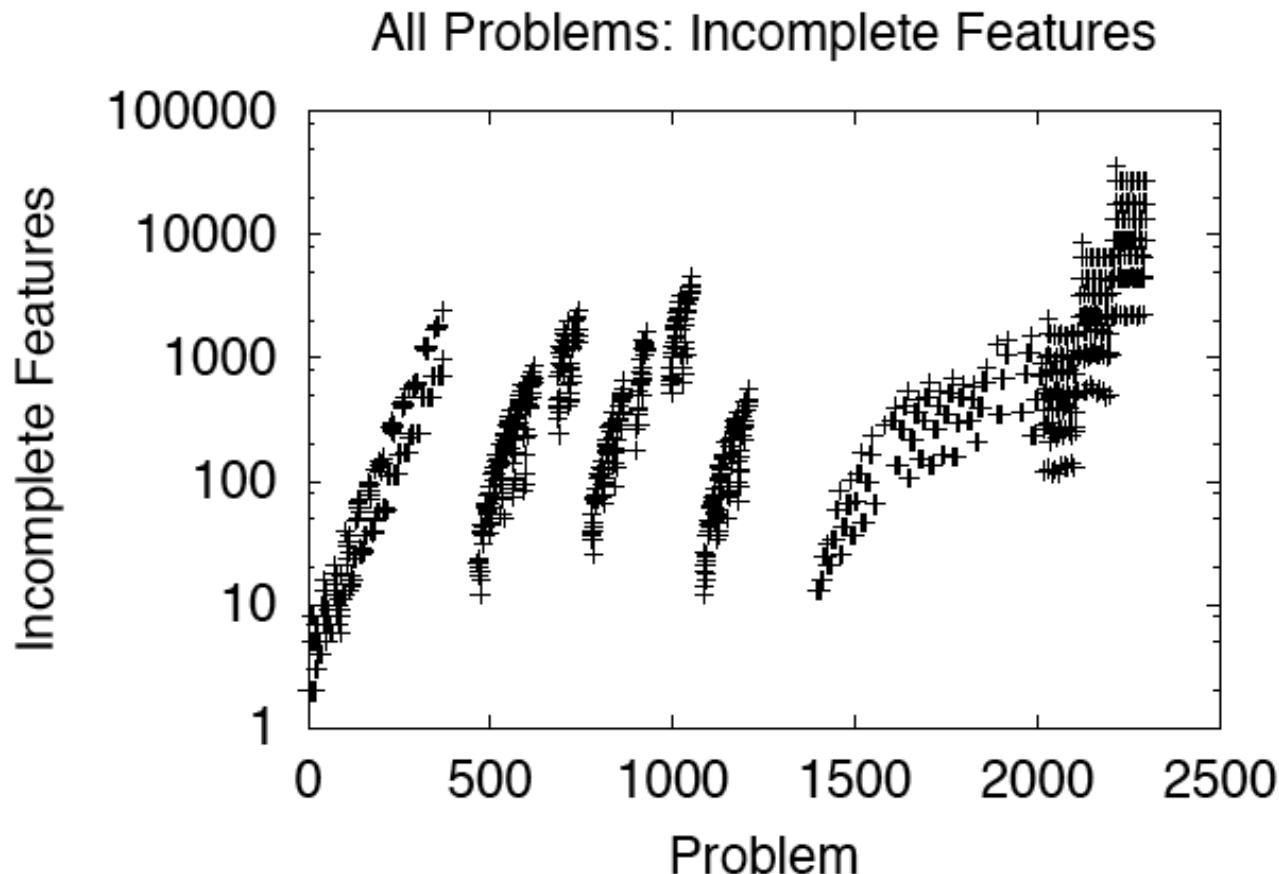
# DeFault Planner

- Dual-Queue GBFS w/ Preferred Operators
- Return First Solution
- All code identical except heuristic functions and search nodes
- Compute plan quality after search
- Pure Java
- JDD BDD package

# Domains



# Incomplete Features



# Questions

- Q1: Does reasoning about incompleteness lead to high quality plans?
- Q2: Does counting prime implicants perform better than counting models?
- Q3: As the number of incomplete features grows, does stronger reasoning about incompleteness help?

# Q1: Quality

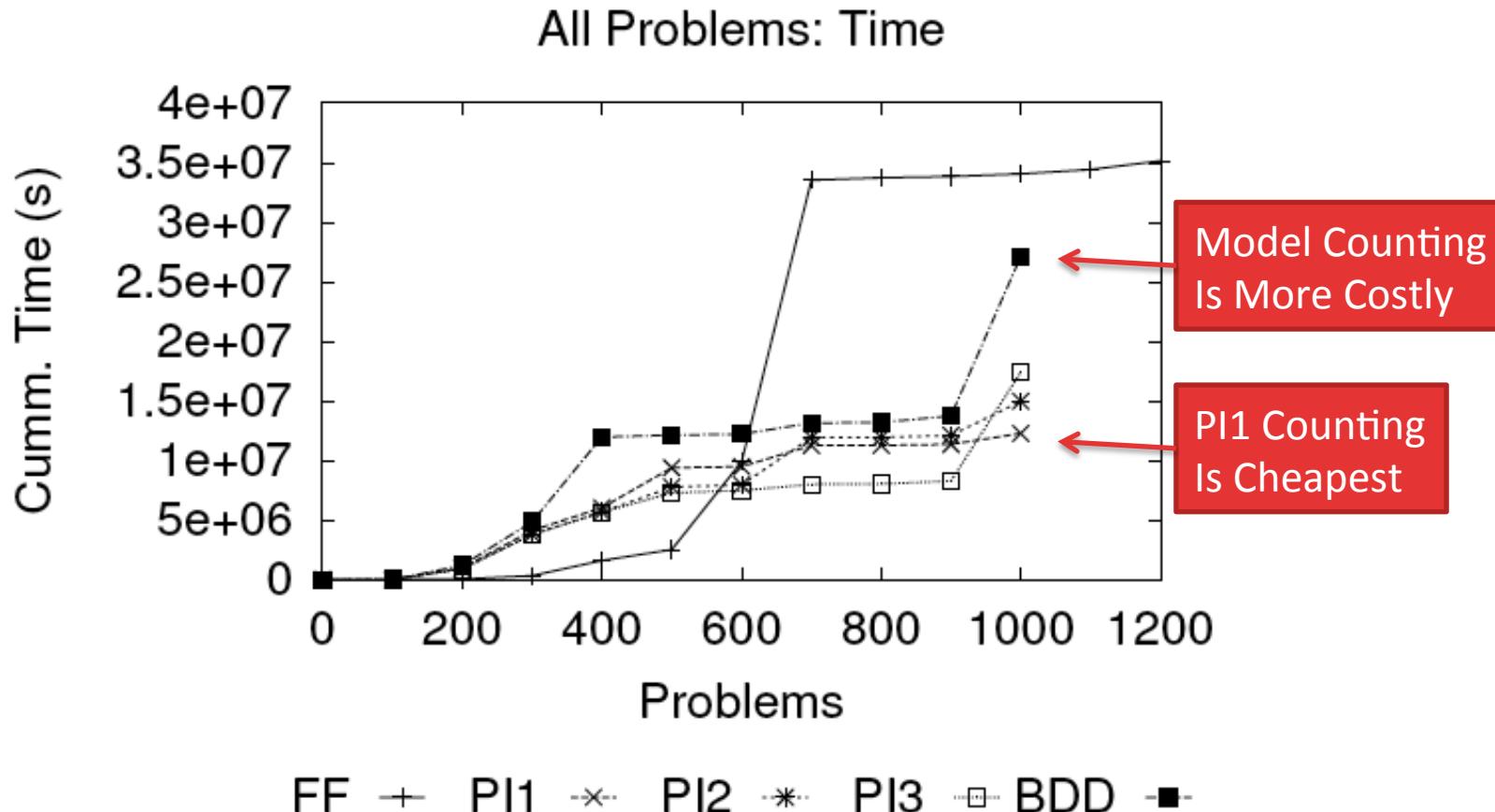
## (# Better Quality)

	FF	PI1	PI2	PI3	BDD
FF	0	148	157	158	121
PI1	<b>603</b>	0	<b>82</b>	<b>80</b>	<b>204</b>
PI2	<b>591</b>	75	0	48	<b>205</b>
PI3	<b>564</b>	59	<b>52</b>	0	<b>194</b>
BDD	<b>494</b>	185	189	186	0

# Q2: Counting Performance (Problems Solved)

Domain	FF	PI1	PI2	PI3	BDD	POND	Domain	FF	PI1	PI2	PI3	BDD	POND
PARCprinter 0.25	130	83	85	<b>86</b>	80	10	Bridges1 0.25	33	<b>19</b>	<b>19</b>	<b>19</b>	<b>19</b>	2
PARCprinter 0.5	130	87	<b>88</b>	87	80	0	Bridges1 0.5	33	<b>15</b>	<b>15</b>	<b>15</b>	<b>15</b>	2
PARCprinter 0.75	130	82	<b>83</b>	81	80	0	Bridges1 0.75	32	<b>18</b>	17	17	<b>18</b>	2
PARCprinter 1.0	13	<b>10</b>	9	9	8	0	Bridges1 1.0	4	<b>2</b>	<b>2</b>	1	<b>2</b>	1
Parcprinter	<b>403</b>	262	<b>265</b>	263	248	10	Bridges2 0.25	29	15	<b>16</b>	<b>16</b>	<b>16</b>	3
Barter 0.25	150	106	128	<b>129</b>	108	60	Bridges2 0.5	31	16	12	12	<b>19</b>	3
Barter 0.5	150	134	<b>137</b>	134	118	45	Bridges2 0.75	31	13	14	15	<b>16</b>	2
Barter 0.75	150	<b>140</b>	138	137	111	27	Bridges2 1.0	4	1	1	0	<b>2</b>	1
Barter 1.0	15	<b>14</b>	<b>14</b>	<b>14</b>	11	2	Bridges3 0.25	36	25	25	25	<b>26</b>	1
Barter	<b>465</b>	394	<b>417</b>	414	348	155	Bridges3 0.5	37	22	22	22	<b>23</b>	2
Pathways 0.25	160	<b>40</b>	<b>40</b>	<b>40</b>	<b>40</b>	19	Bridges3 0.75	38	<b>25</b>	<b>25</b>	24	<b>25</b>	1
Pathways 0.5	160	<b>70</b>	60	50	60	13	Bridges3 1.0	4	<b>3</b>	<b>3</b>	<b>3</b>	2	1
Pathways 0.75	170	<b>60</b>	50	40	<b>60</b>	12	Bridges	<b>312</b>	174	171	169	<b>183</b>	21
Pathways 1.0	19	5	6	6	<b>7</b>	2	Total	1689	1005	<b>1009</b>	982	946	232
Pathways	<b>509</b>	<b>175</b>	156	136	167	46							

# Q2: Counting Performance (Cumulative Solving Time)



# Q3: Counting and Features (Problems Solved)

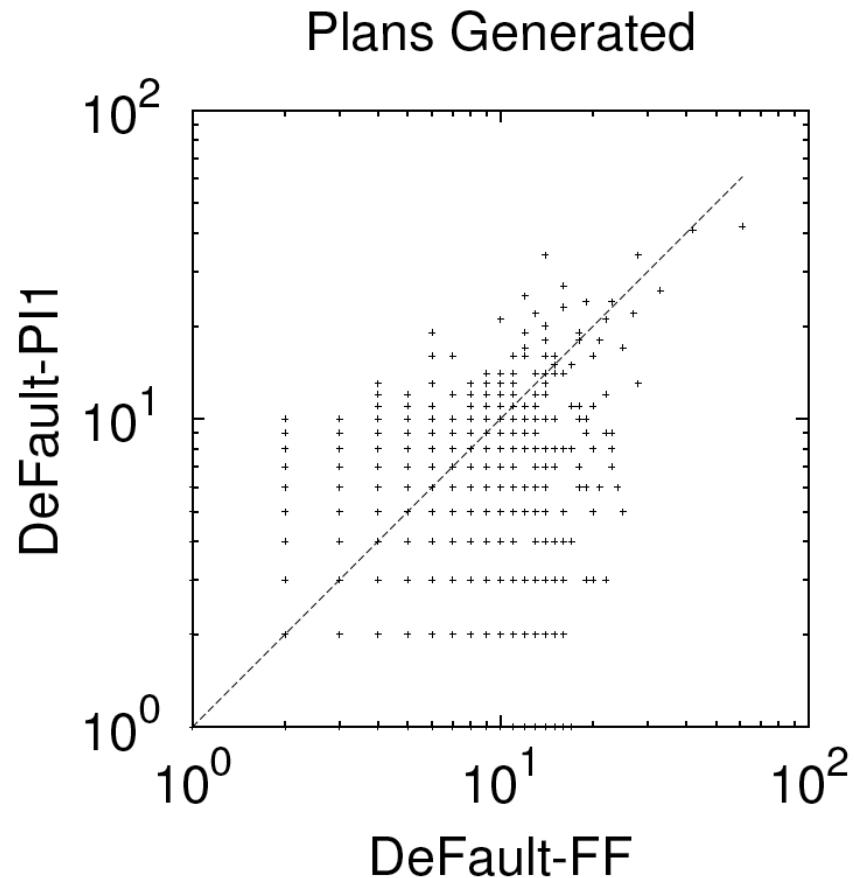
Domain	FF	PI1	PI2	PI3	BDD	POND
PARCprinter 0.25	130	83	85	<b>86</b>	80	10
PARCprinter 0.5	130	87	<b>88</b>	87	80	0
PARCprinter 0.75	130	82	<b>83</b>	81	80	0
PARCprinter 1.0	13	<b>10</b>	9	9	8	0
Parcprinter	403	262	<b>265</b>	263	248	10
Barter 0.25	150	106	128	<b>129</b>	108	60
Barter 0.5	150	134	<b>137</b>	134	118	45
Barter 0.75	150	<b>140</b>	138	137	111	27
Barter 1.0	15	<b>14</b>	<b>14</b>	<b>14</b>	11	2
Barter	465	394	<b>417</b>	414	348	155
Pathways 0.25	160	<b>40</b>	<b>40</b>	<b>40</b>	<b>40</b>	19
Pathways 0.5	160	<b>70</b>	60	50	60	13
Pathways 0.75	170	<b>60</b>	50	40	<b>60</b>	12
Pathways 1.0	19	5	6	6	7	2
Pathways	509	<b>175</b>	156	136	167	46

Domain	FF	PI1	PI2	PI3	BDD	POND
Bridges1 0.25	33	<b>19</b>	<b>19</b>	<b>19</b>	<b>19</b>	2
Bridges1 0.5	33	<b>15</b>	<b>15</b>	<b>15</b>	<b>15</b>	2
Bridges1 0.75	32	<b>18</b>	17	17	<b>18</b>	2
Bridges1 1.0	4	<b>2</b>	<b>2</b>	1	<b>2</b>	1
Bridges2 0.25	29	15	<b>16</b>	<b>16</b>	<b>16</b>	3
Bridges2 0.5	31	16	12	12	<b>19</b>	3
Bridges2 0.75	31	13	14	15	<b>16</b>	2
Bridges2 1.0	4	1	1	0	<b>2</b>	1
Bridges3 0.25	36	25	25	25	<b>26</b>	1
Bridges3 0.5	37	22	22	22	<b>23</b>	2
Bridges3 0.75	38	<b>25</b>	<b>25</b>	24	<b>25</b>	1
Bridges3 1.0	4	<b>3</b>	<b>3</b>	<b>3</b>	2	1
Bridges	312	174	171	169	<b>183</b>	21
Total	1689	1005	<b>1009</b>	982	946	232

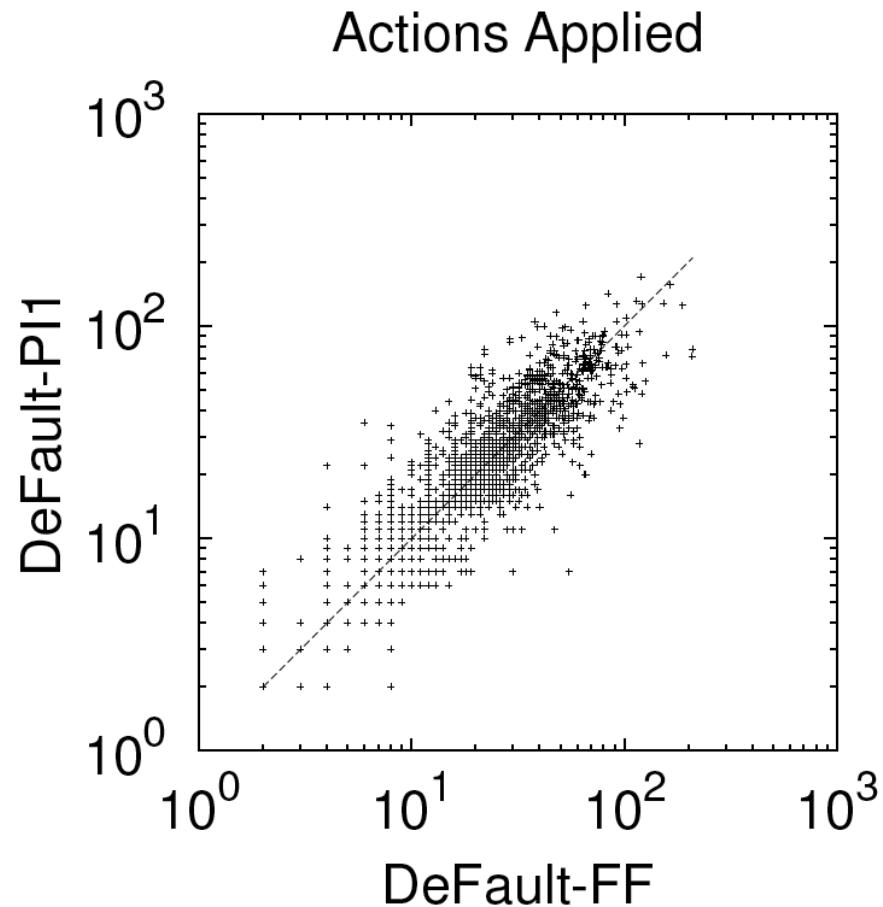
# Execution in Incomplete Domains

- [Chang and Amir, ICAPS06]
  - Plan optimistically much like a classical planner
  - Learn adds and dels
  - Emphasizes Compact Filtering
- Goalie
  - Plan with DeFault
  - Learn pre, add, and dels
  - Emphasizes Robust Planning
- Q4: Does reasoning about incompleteness reduce the number of execution failures during execution?

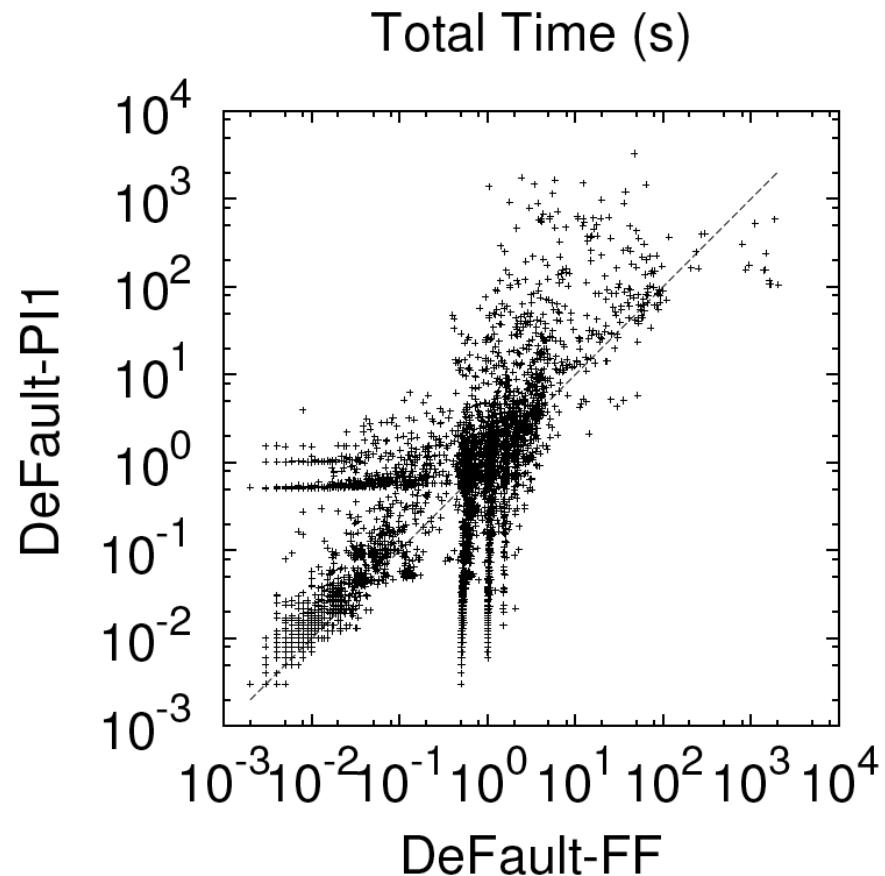
# Q4: Planner Invocations



# Actions Applied



# Total Time



# Summary

- First approach to planning in incomplete STRIPS domains
- Counting diagnoses (PIs) is faster than model counting and returns comparable quality plans
- Executing robust Incomplete plans requires less re-planning

# Future Work

- More types of models
  - Conditional Effects, Temporal, First-Order, ...
- Correlated Incompleteness
- Priors
- Alternative Representations of Failure Expl.
- Approximate Belief States
- Knowledge Acquisition

# Thanks

DeFault and Goalie Java source code and domain  
generators available online

<http://www.cs.usu.edu/~danbryce/software/default.jar>