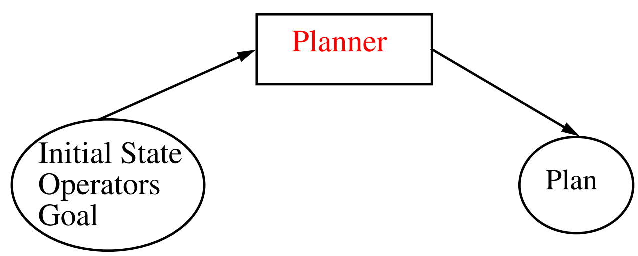
Planning as Satisfiability

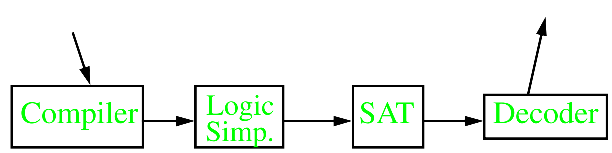
Todd Millstein
todd@cs

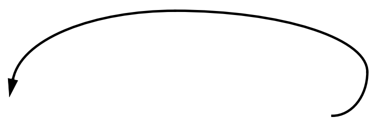
October 26, 1998

Planning



as Satisfiability





Brief History

- **Kautz, Selman 1992**
  - first described planning as satisfiability
- **Kautz, Selman 1996**
  - showed remarkable performance of the method
  - hand encodings, domain-specific info
- **Kautz, McAllester, Selman 1996**
  - discussed several encodings

Brief History (cont.)

- **Weld 1996**
  - imposed his love for SATplan on CSE573 students
- **Ernst, Millstein, Weld 1997**
  - parametrized space of encodings
  - first automatic compiler
- **Kautz, Selman 1998**
  - Blackbox compiler
  - study of domain-specific info

Actions and Time

Paint(b, c)
Precond: Block(b) ∧ Holding(b) ∧ Color(c)
Effect: Painted(b) ∧ HasColor(b, c)

Time	Layer
0	Initial State (State 0)
1	Action(s) 1
2	State 1
...	...
2n	State n (contains Goal)

fluents: HasColor-A-Red-0, HasColor-A-Red-2, ...

The Axioms

Axiom	Description/Example
INIT	The initial state holds at time 0
GOAL	The goal holds at time 2n
A⇒P,E	Paint(A,Red,t) ⇒ Block(A,t-1) Paint(A,Red,t) ⇒ HasColor(A,Red,t+1)
FRAME	Classical Explanatory
AT-LEAST-ONE	(Act1 ∨ Act2 ∨ Act3 ∨...)
EXCLUDE	(~Act1 ∨ ~Act2)

The Space of Encodings

Action Representation:
- how to represent actions as propositional vars.
  - Regular
  - Simply-split
  - Overloaded-split
  - Bitwise

Frame Clauses:
- ensure unaffected fluents don't change
  - Classical
  - Explanatory

Action Representation

Representation	One Propositional Variable per	Example
Regular	fully-instantiated action	Paint-A-Red, Paint-A-Blue, Move-A-Table

Simply-split	fully-instantiated action's argument	Paint-Arg1-A ∧ Paint-Arg2-Red

Overloaded-split	fully-instantiated argument	Act-Paint ∧ Arg1-A ∧ Arg2-Red

Bitwise	binary encoding of actions	Bit1 ∧ ~Bit2 ∧ Bit3

Paint-A-Red = 5

more vars

more clses

Frame Clauses

Frame Clause	States how fluents...	Example	Requires Additional Axioms
Classical	stay the same	HasColor(A,Blue,t-1) ∧ Paint(B,Red,t) ⇒ HasColor(A,Blue,t+1)	AT-LEAST-ONE

Explanatory	change	HasColor(A,Blue,t-1) ∧ ~HasColor(A,Blue,t+1) ⇒ Paint(A,Red,t) ∨ Paint(A,Grey,t) ∨...	EXCLUDE ~A1 ∨ ~A2

Eight Base Encodings

Action Representation

	Regular	Simple	Overloaded	Bitwise
Classical Frames	AT-LEAST-ONE	AT-LEAST-ONE	AT-LEAST-ONE	
Explanatory Frames	EXCLUDE	EXCLUDE	EXCLUDE	

Optimization 1: Factored Splitting

~HasColor-A-Blue-(t-1) ∧ Paint-Arg1-B-t ∧
Paint-Arg2-Red-t ⇒ ~HasColor-A-Blue-(t+1)

~HasColor-A-Blue-(t-1) ∧ Paint-Arg1-B-t ∧
Paint-Arg2-Grey-t ⇒ ~HasColor-A-Blue-(t+1)

...

becomes

~HasColor-A-Blue-(t-1) ∧ Paint-Arg1-B-t ∧
⇒ ~HasColor-A-Blue-(t+1)

Optimization 2: Types

A *type* is a fluent which no action affects.
e.g. Block(x), Color(x)

Paint(b, c)
Precond: Holding(b) ∧ ~Painted(b) ∧ Color(c)
Effect: Painted(b) ∧ HasColor(b, c)

PickUp(b)
Precond: Block(b) ∧ ArmEmpty
Effect: Holding(b) ∧ ~ArmEmpty

More General Questions

- What about using a stochastic, special-purpose planner, getting the best of both worlds?

- What other problems would be amenable to reduction to satisfiability?

Comparisons Among Encodings

- **Explanatory frames are superior to classical**
  - few actions affect each fluent
  - explanatory frames allow parallelism
- **Parallelism is a major factor**
  - fewer mutual exclusion clauses
  - fewer time steps
- ***Regular action representation is smallest***
  - exploits full parallelism
  - aids simplification
- **Overloaded, bitwise reps. are infeasible**
  - prohibitively many clauses
  - sharing hinders simplification

Domain-Specific Axioms

Adding domain-specific axioms decreases solve time dramatically.
- only one block can be held at a time
- no block should be painted more than once

$\frac{\text{domain info}}{\text{no domain info}}$

bw-large	Vars	Clauses	Time
a	.86	1.53	.26
b	.88	1.84	.38
c	.86	2.24	<.05

Directions for Future Work

- Inference of domain-specific axioms
- How is size related to solve time?
- What other factors determine solve time?
- Hybrid encodings