SAT $\cap$ AI

Henry Kautz
University of Rochester

Outline

- Ancient History: Planning as Satisfiability
- The Future: Markov Logic

Part I

- Ancient History: Planning as Satisfiability
  - Planning
  - SAT encoding
  - 3 good ideas:
    - Parallel actions
    - Plan graph pruning
    - Transition based encoding

Planning

- Find a plan that transform an initial state to a goal state
  - What is a state?
  - What is a plan?

Classic Planning

- Find a sequence of actions that transform an initial state to a goal state
  - State = complete truth assignment to a set of time-dependent propositions (fluents)
  - Action = a partial function State $\rightarrow$ State
- Fully observed, deterministic

STRIPS

- Set of possible actions specified by parameterized operator schemas and (typed) constants

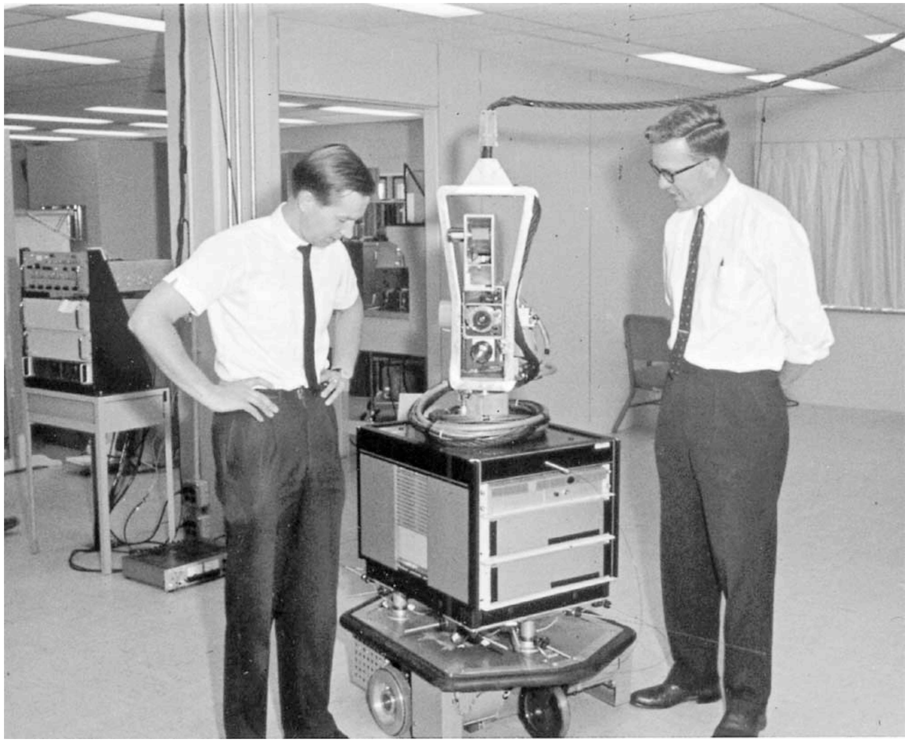
  operator: Fly(a,b)
  precondition: At(a), Fueled
  effect: At(b), ~At(a), ~Fueled

  constants: {NY, Boston, Seattle}

- Fluents not mentioned in effect are unchanged by action

STRIPS

- Introduced for Shakey the robot (1969)
  - Generate plan
  - Start executing
  - Sense state after each action, verifying it is as expected
  - If not, stop and replan

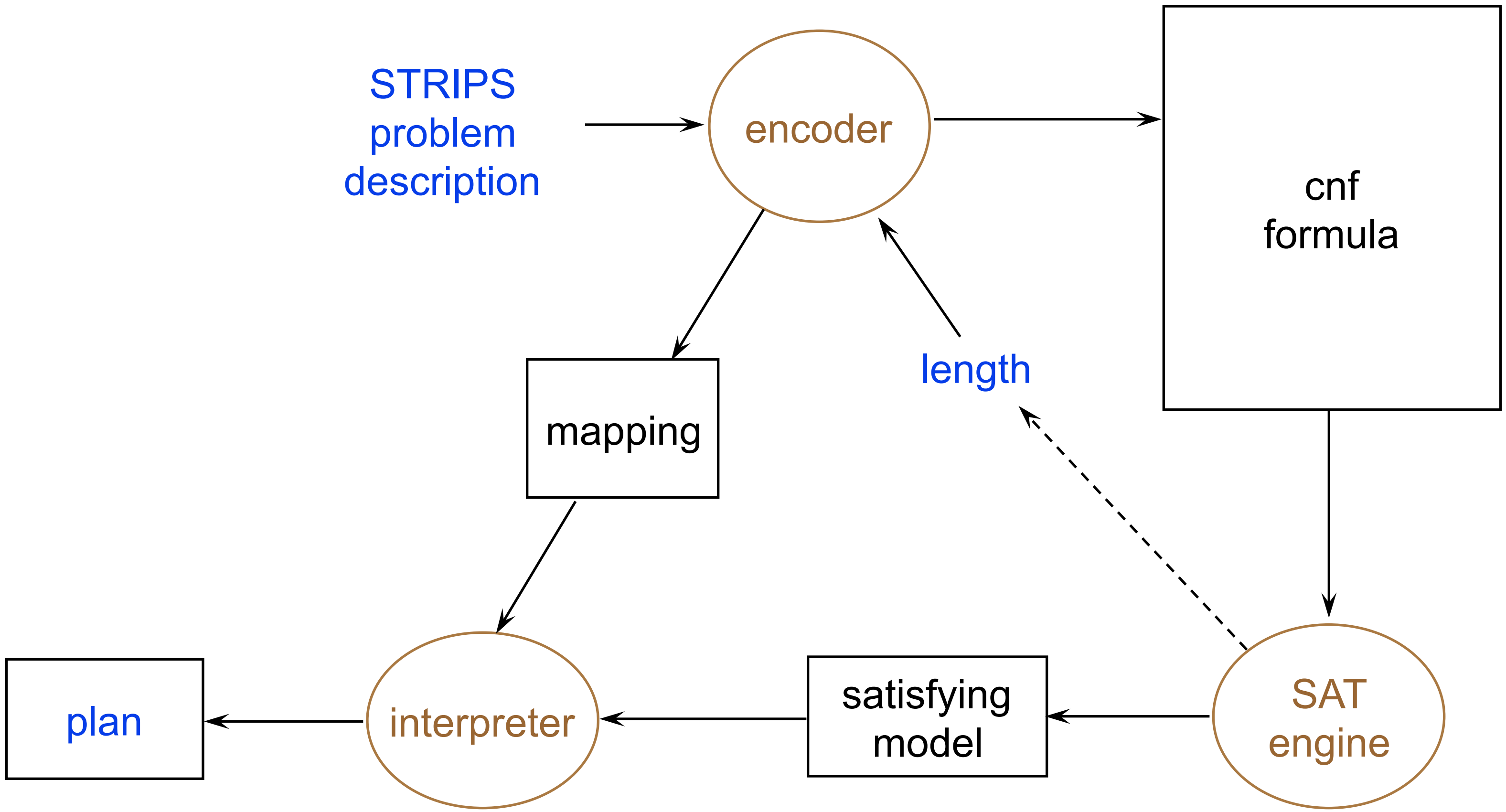


- Still a widely-used method for robot control (vs. POMDP *etc*)

STRIPS

- Complexity
  - Unbounded length: PSPACE-complete
  - Bounded length: NP-complete
- Algorithms
  - Backward chaining on subgoals (1969)
  - Search in space of partially-order plans (1987)
  - Planning as satisfiability (1992, 1996)
  - Graphplan (1996)
  - Forward- chaining heuristic search (1999)

SATPLAN


STRIPS
problem
description
encoder
cnf
formula
mapping
length
plan
interpreter
satisfying
model
SAT
engine

Clause Schemas

$$\forall x \in \{A, B, C\} P(x)$$

represents

$$P(A) \wedge P(B) \wedge P(C)$$

$$\exists x \in \{A, B, C\} P(x)$$

represents

$$P(A) \vee P(B) \vee P(C)$$

SAT Encoding

- Time is sequential and discrete
  - Represented by integers
  - Actions occur instantaneously at a time point
  - Each fluent is true or false at each time point
- If an action occurs at time i, then its preconditions must hold at time i
- If an action occurs at time i, then its effects must hold at time i+1
- If a fluent changes its truth value from time i to time i+1, one of the actions with the new value as an effect must have occurred at time i
- Two actions cannot occur at the same time
- The initial state holds at time 0, and the goals hold at a given final state K

SAT Encoding

- If an action occurs at time i, then its preconditions must hold at time i

operator: Fly(p,a,b)

precondition: At(p,a), Fueled(p)

effect: At(p,b), ~At(p,a), ~Fueled (p)

Constant types: *Times, Planes, Cities*

$$\forall i \in Times$$
$$\forall p \in Planes$$
$$\forall a \in Cities$$
$$\forall b \in Cities$$
$$\text{fly}(p,a,b,i) \supset (\text{at}(p,a,i)) \wedge \text{fuel}(p,i))$$

SAT Encoding

- If an action occurs at time i, then its effects must hold at time i+1

operator: Fly(p,a,b)

precondition: At(p,a), Fueled(p)

effect: At(p,b), ~At(p,a), ~Fueled (p)

Constant types: *Times, Planes, Cities*

$$\forall i \in Times$$
$$\forall p \in Planes$$
$$\forall a \in Cities$$
$$\forall b \in Cities$$
$$\text{fly}(p,a,b,i) \supset (\text{at}(p,b,i+1)) \wedge \neg \text{at}(p,a,i+1) \wedge \neg \text{fuel}(p,i+1))$$

SAT Encoding

- If a fluent changes its truth value from time i to time i+1, one of the actions with the new value as an effect must have occurred at time i
- Change from false to true

operator: Fly(p,a,b)

precondition: At(p,a), Fueled(p)

effect: At(p,b), ~At(p,a), ~Fueled (p)

Constant types: *Times, Planes, Cities*

$$\forall i \in Times$$
$$\forall p \in Planes$$
$$\forall b \in Cities$$
$$(\neg \mathrm{at}(p,b,i) \wedge \mathrm{at}(p,b,i+1)) \supset$$
$$\exists a \in Cities \;.\; \mathrm{fly}(p,a,b,i)$$

SAT Encoding

- If a fluent changes its truth value from time i to time i+1, one of the actions with the new value as an effect must have occurred at time i
- Change from true to false:

$$\forall i \in Times$$
$$\forall p \in Planes$$
$$\forall a \in Cities$$
$$(\mathrm{at}(p,a,i) \wedge \neg \mathrm{at}(p,a,i+1)) \supset$$
$$\exists b \in Cities \;.\; \mathrm{fly}(p,a,b,i)$$

operator: Fly(p,a,b)

precondition: At(p,a), Fueled(p)

effect: At(p,b), ~At(p,a), ~Fueled (p)

Constant types: *Times, Planes, Cities*

Action Mutual Exclusion

- Two actions cannot occur at the same time

operator: Fly(p,a,b)

precondition: At(p,a), Fueled(p)

effect: At(p,b), ~At(p,a), ~Fueled (p)

Constant types: *Times, Planes, Cities*

$$\forall i \in Times$$
$$\forall p1, p2 \in Planes$$
$$\forall a, b, c, d \in Cities$$
$$\neg \mathrm{fly}(p1,a,b,i) \vee \neg \mathrm{fly}(p2,c,d,i)$$

Result

- 1992: can find plans with 5 actions
  - Typical for planners at that time...
- 1996: finds plans with 60+ actions
- What changed?
  - Better SAT solvers
  - Two good ideas:
    - Parallel actions
    - Plan graph pruning

Parallel Actions

- Allow multiple actions to occur at the same time step if they are non-interfering:
  - *Neither negates a precondition or effect of the other*
- Can greatly reduce solution horizon in many domains

$$\begin{aligned}&\forall i \in Times\\ &\quad \forall p1, p2 \in Planes\\ &\quad\quad \forall a,b,c,d \in Cities\\ &\quad\quad\quad \neg \mathrm{fly}(\mathrm{p1,a,b,i}) \vee \neg \mathrm{fly}(\mathrm{p2,c,d,i})\end{aligned} \longrightarrow \begin{aligned}&\forall i \in Times\\ &\quad \forall p \in Planes\\ &\quad\quad \forall a,b,c,d \in Cities\\ &\quad\quad\quad \neg \mathrm{fly}(\mathrm{p,a,b,i}) \vee \neg \mathrm{fly}(\mathrm{p,c,d,i})\end{aligned}$$

Graph Plan

- Graphplan (Blum & Furst 1996) introduced a new planning algorithm:
  - Instantiate a "plan graph" in a forward direction
    - Nodes: ground facts and actions
    - Links: supports and mutually-exclusive
  - Each level of the graph contains all the reachable propositions at that time point
    - Set of propositions, not a set of states!
  - Seach for a subset of the graph that
    - Supports all the goal propositions
    - Contains no mutually-exclusive propositions

Initial State

facts actions facts actions

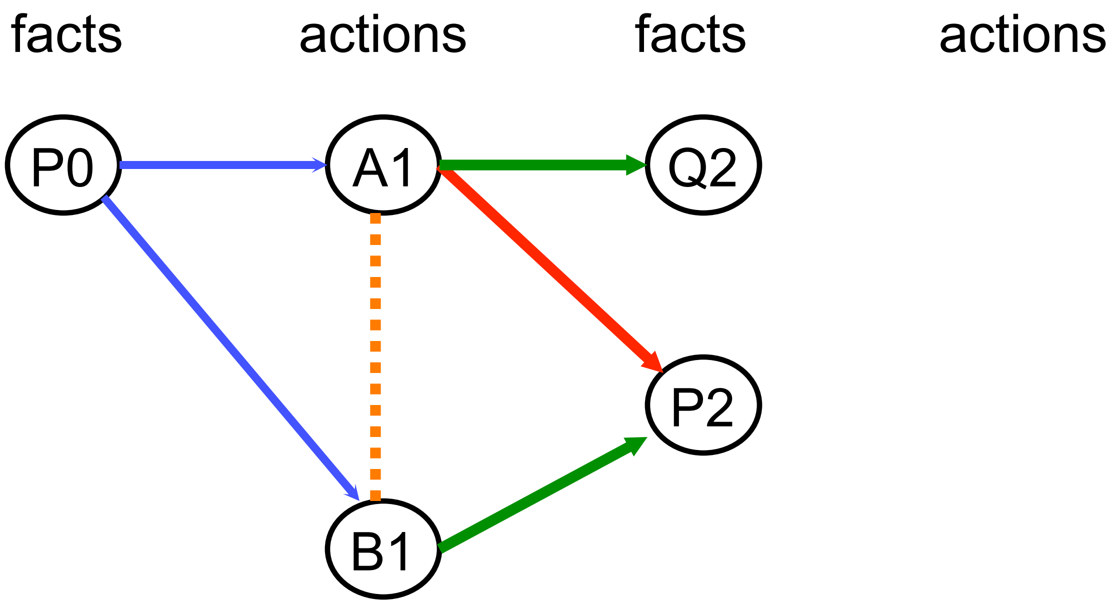
P1

action a: pre p; effect ~p, q

action b: pre p; effect p

action c: pre p, q; effect r

Growing Next Level

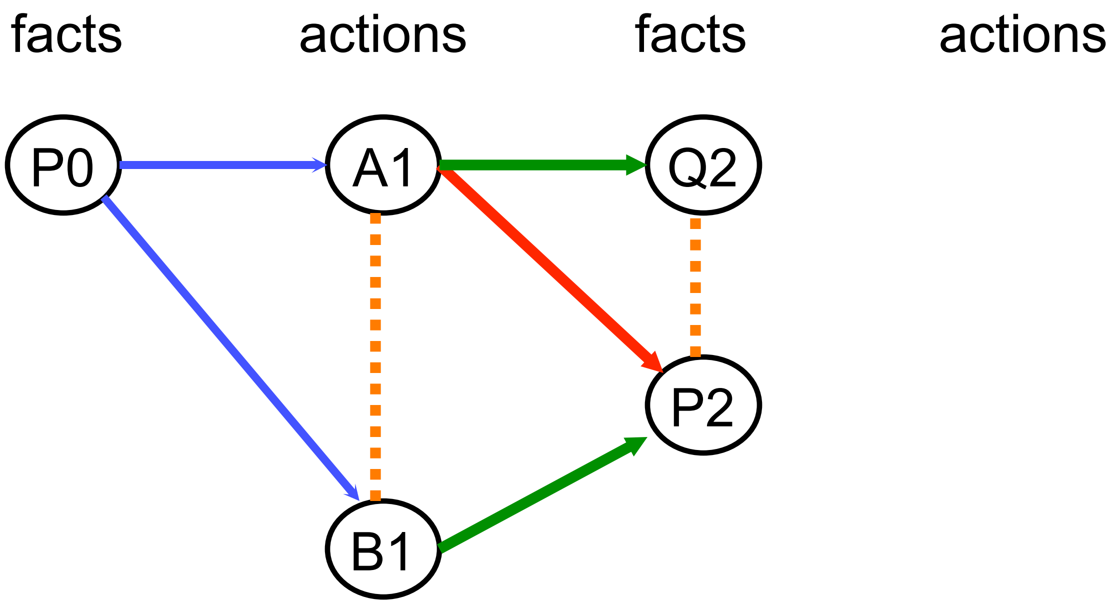


action a: pre p; effect ~p, q

action b: pre p; effect p

action c: pre p, q; effect r

Propagating Mutual Exclusion

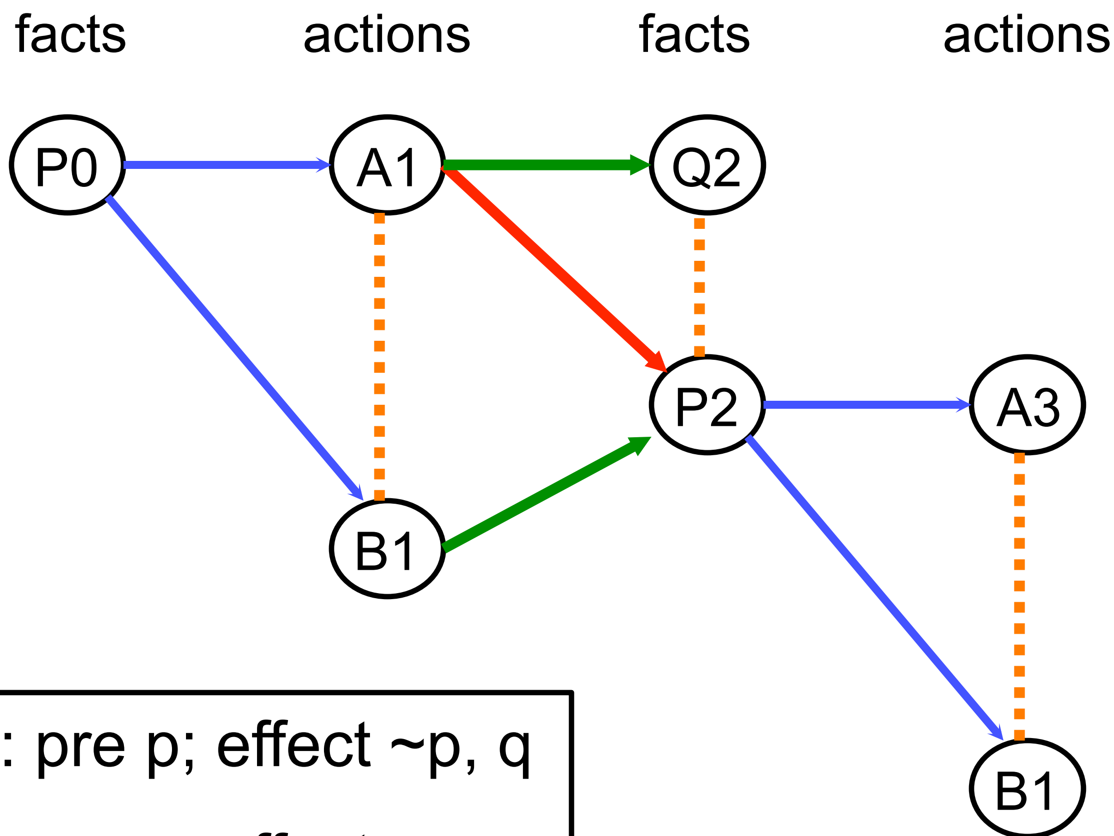


action a: pre p; effect ~p, q

action b: pre p; effect p

action c: pre p, q; effect r

Growing Next Level



action a: pre p; effect ~p, q

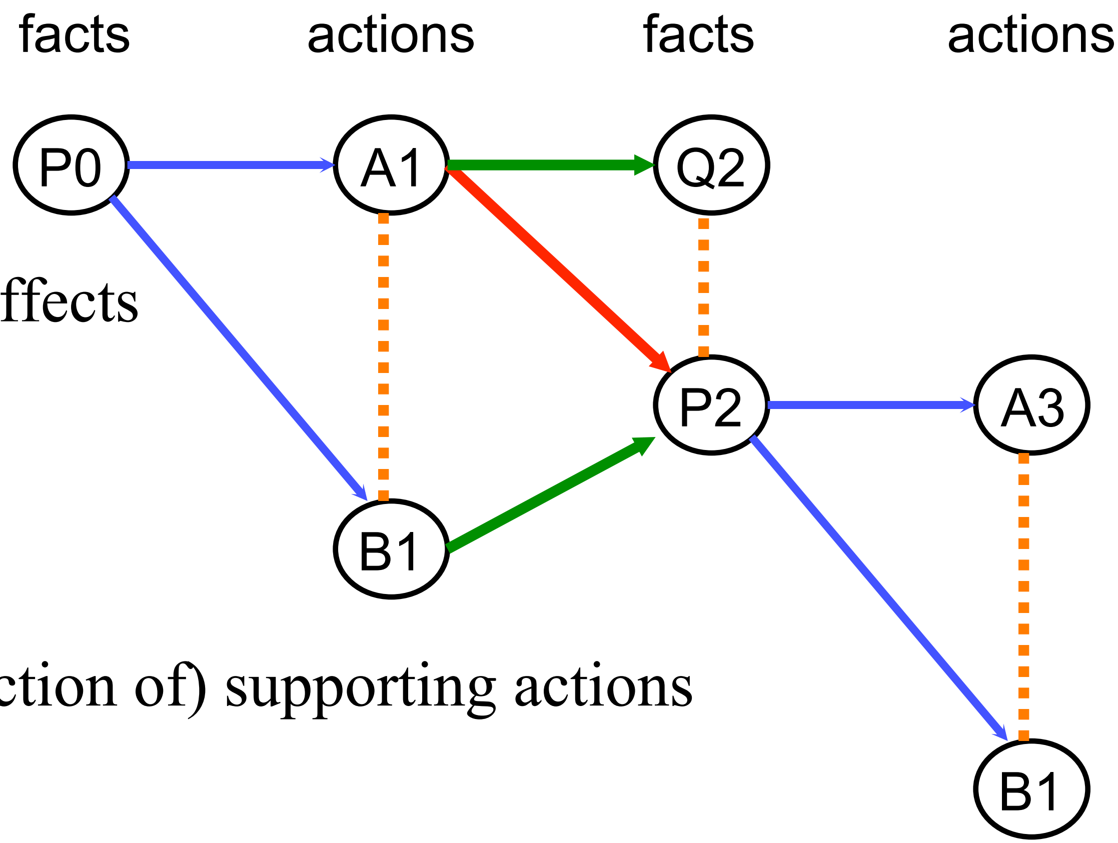
action b: pre p; effect p

action c: pre p, q; effect r

Plan Graph Pruning

- The SATPLAN encoding (with parallel actions) can be directly created from the plan graph
- Prunes many unnecessary propositions and clauses
- “Propagated mutexes” may or may not be included in the translation
  - Logically redundant
  - May help or hinder particular SAT solvers

Translation to SAT



Actions imply

preconditions and effects

$a1 \rightarrow p0$

$a1 \rightarrow q2$

$a1 \rightarrow \neg q2$

Facts imply (disjunction of) supporting actions

$q2 \rightarrow a1$

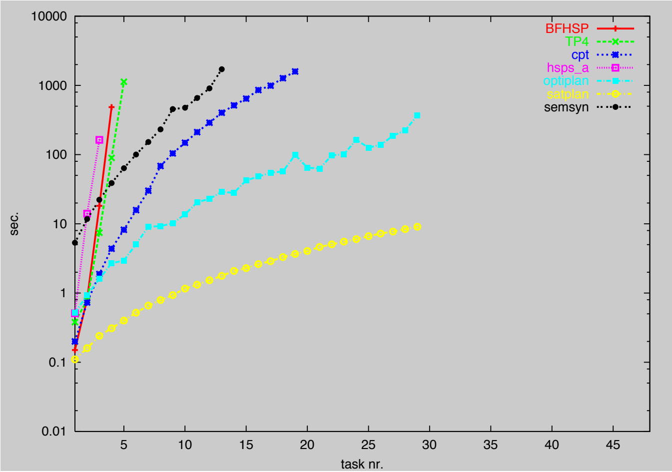
$p2 \rightarrow b1$

Mutual exclusions

$\neg a1 \vee \neg b1$

Blast From the Past

Performance



- SATPLAN and variants won optimal deterministic STRIPS tracks of International Planning Competition through 2006
  - 10 year run – steady performance improvements due to SAT solvers
- 2008: Change in rules: optimality defined as function of action and resource costs, not parallel time horizon
  - Opportunity for SMT (see Hoffmann et al 2007)

Transition-Based Encodings

- Surprisingly few new ideas for encodings
- One good one: transition-based encodings (Huang, Chan, Zhang 2010)
  - Based on a double-reformulation of STRIPS:
  - Represent states in terms of *multi-valued variables* (SAS+)
  - Encode *transitions* in the state variables as the SAT propositions

SAS+ Representation

Strips

AT pkg loc1

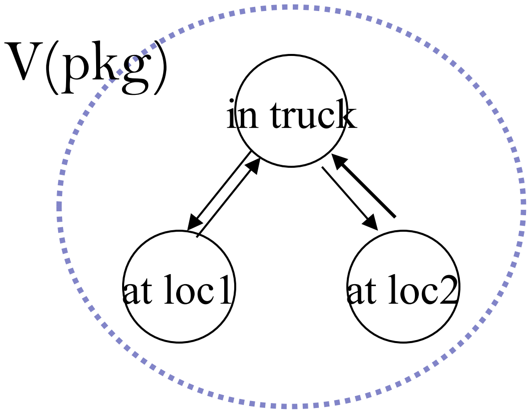
AT pkg loc2

IN pkg truck

AT truck loc1

AT truck loc2

SAS+

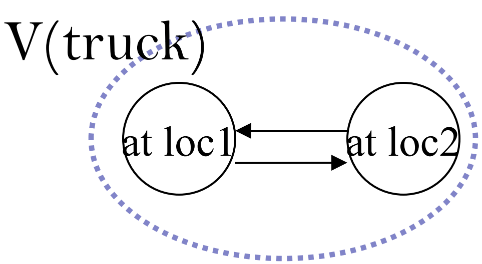


pkg:loc1→truck

pkg:truck→loc1

pkg:truck→loc2

pkg:loc2→truck



Transition: Change between values in a multi-valued variable

Comparison of STRIPS and SAS+

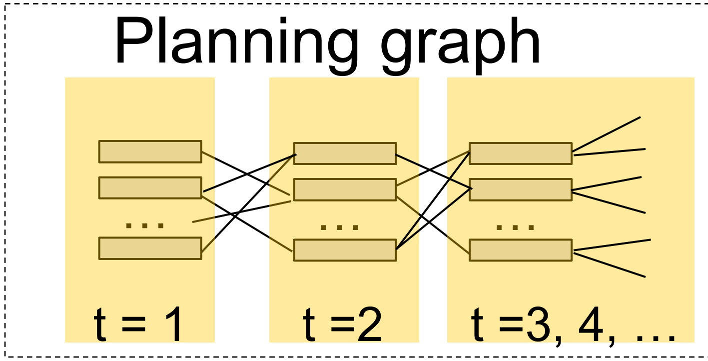
	STRIPS		SAS+
Definition	a set of preconditions, a set of add effects, a set of delete effects		A set of transitions
Example	**(LOAD pkg truck loc1)**		
	Pre:	(at truck loc1), (at pkg loc1)	pkg:(loc1→truck) truck: (loc1→loc1)
	Del:	(at pkg loc1)	
	Add:	(in pkg truck)	

Usually there are ***fewer*** transitions than actions

Hierarchical relationships between actions and transitions

Overview of New Encoding

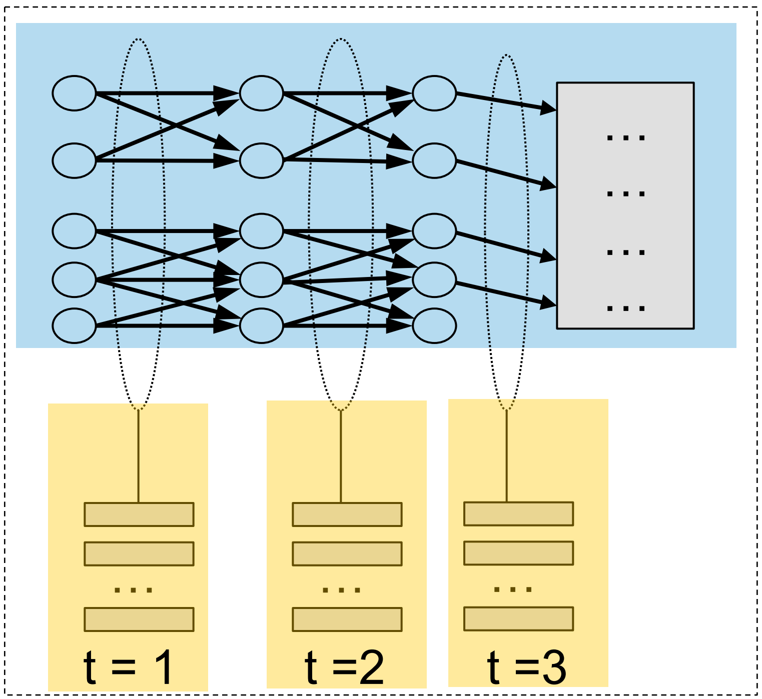
Actions



SAT Instance:
Facts and actions

Strips Based Encoding

Transitions



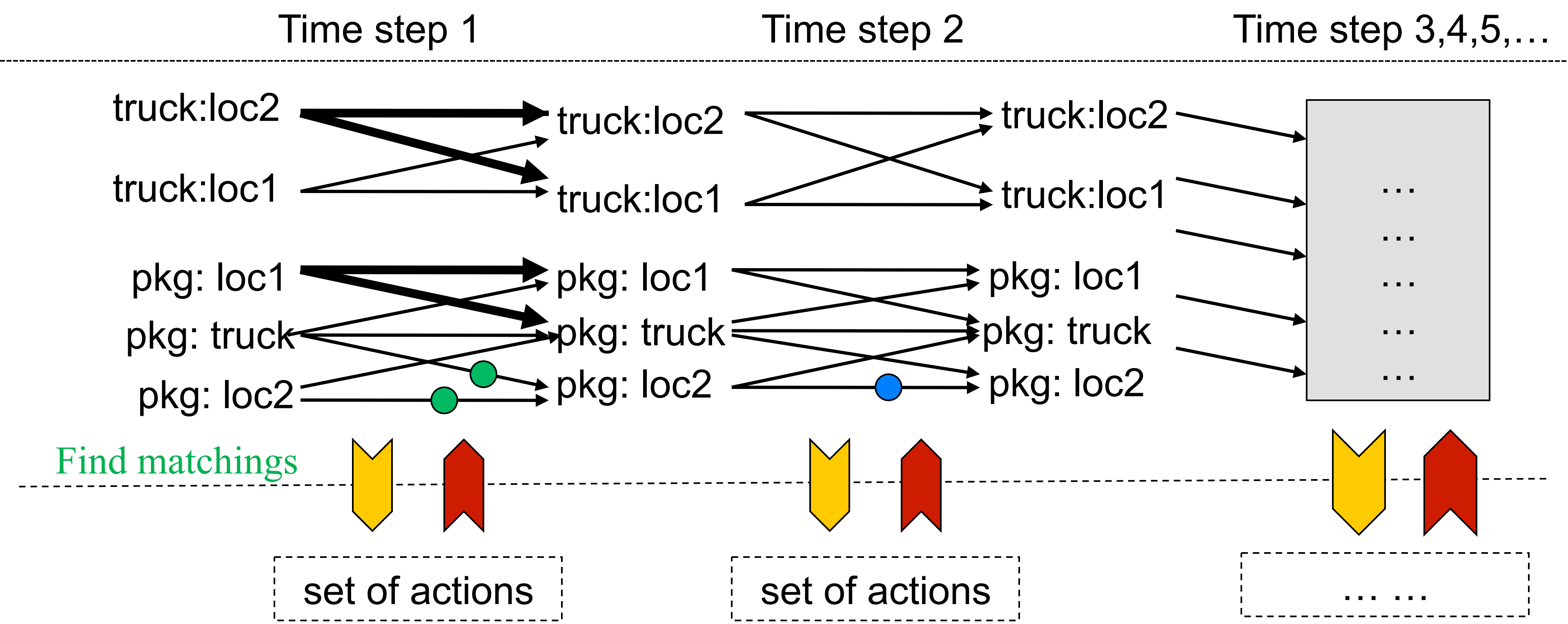
Actions

SAT Instance (Part 1):
transitions

SAT Instance (Part 2):
matching actions and transitions (multiple independent ones)

SAS+ Based New Encoding

Clauses in New Encoding, Example



1. Progression of transitions over time steps (blue one implies green ones)
2. Initial state and goal (Bold ones)
3. Matching actions and transitions
4. Action mutual exclusions and transition mutual exclusions

Clauses for Action-Transition Matching

Actions:

x, y, z

Transitions:

a, b, c, d

x: {a, b, c}

y: {b, c, d}

z: {a, c, d}

- Action implies transitions:

$$x_t \rightarrow (a_t \wedge b_t \wedge c_t)$$

$$y_t \rightarrow (b_t \wedge c_t \wedge d_t)$$

$$z_t \rightarrow (a_t \wedge c_t \wedge d_t)$$

- Transition implies actions:

$$a_t \rightarrow (x_t \vee z_t)$$

$$b_t \rightarrow (x_t \vee y_t)$$

$$c_t \rightarrow (x_t \vee y_t \vee z_t)$$

$$d_t \rightarrow (y_t \vee z_t)$$

- Action mutual exclusions:

$$x_t \rightarrow \neg y_t ; y_t \rightarrow \neg z_t ; z_t \rightarrow \neg x_t$$

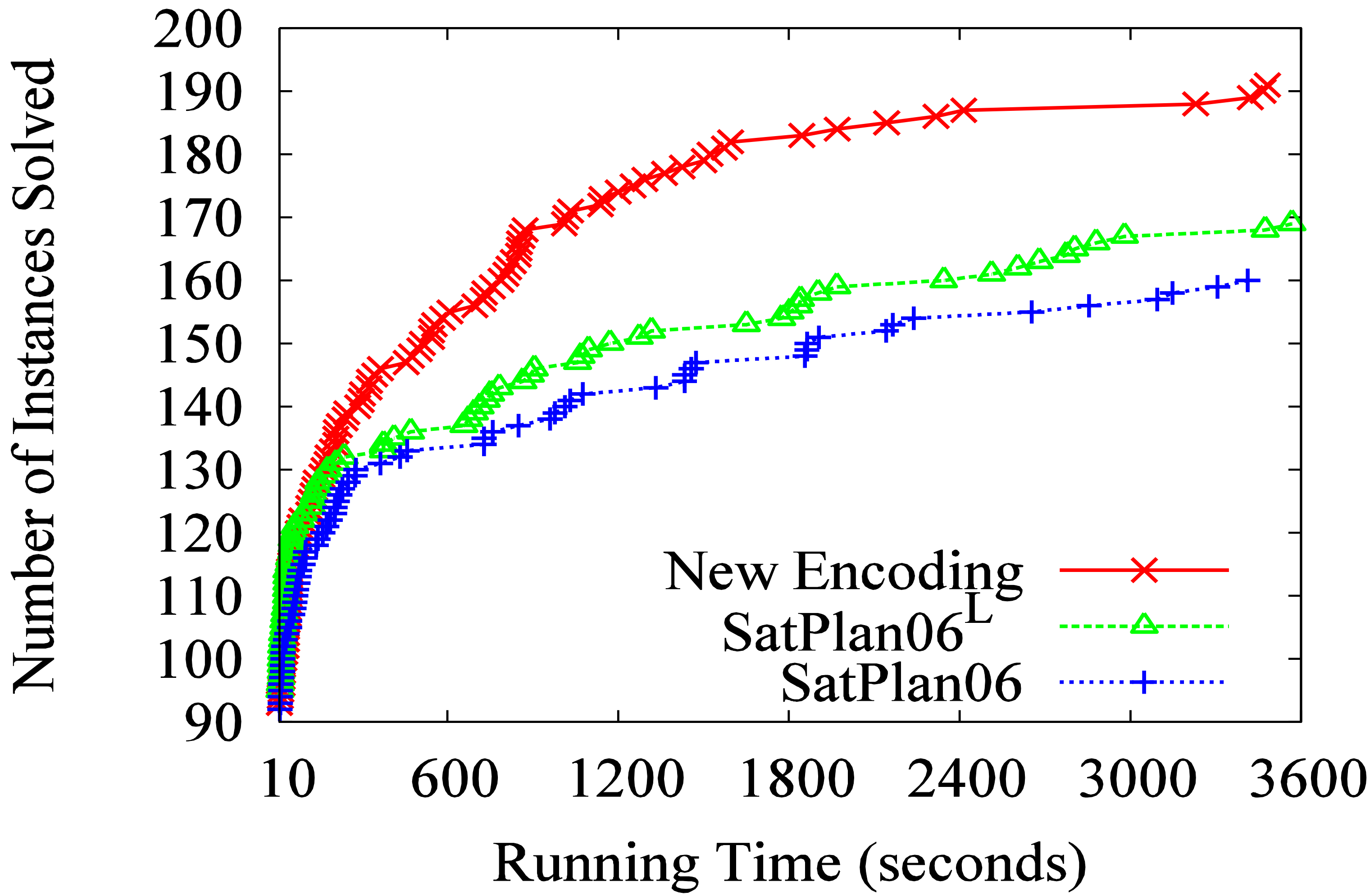
These clauses repeat in each time step ***t***.

Strips v.s. SAS+ Based Encodings

	Strips	SAS+
Variables	❑ Actions and Facts	❑ Actions and Transitions
Clauses	❑ Logics of actions across time steps, subject to initial state and goal ($O((2^A)^N)$)	❑ Logics of transitions across time steps, subject to initial state and goal ($O((2^T)^N)$) ***T* is much smaller than *A*** ❑ Logics of finding a matching action set for transitions, in each time step t (K) ***N* small independent matching problems** *Exact Cover problem*[Karp72]
	Worst case state space size: $O((2^A)^N)$	Worst case state space size: $O((2^T)^N NK)$

N, T, A: number of time steps, transitions and actions

Number of Solvable Instances versus Time Limits



Better performances in 10 domains out of 11 tested (from IPC3,4,5)

Detailed Results

	SatPlan06				New Encoding			
Instances	Time (sec)	#Variables	#Clauses	Size (MB)	Time	#Variables	#Clauses	Size
Airport40	2239.4	327,515	13,206,595	807	583.3	396,212	3,339,914	208
Driverslog17	2164.8	61,915	2,752,787	183	544.1	74,680	812,312	56
Freecell4	364.3	17582	6,114,100	392	158.4	26,009	371,207	25
Openstack4	212.1	3,709	66,744	5	33.6	4,889	20,022	2
Pipesworld12	3147.3	30,078	13,562,157	854	543.7	43,528	634,873	44
TPP30	3589.7	97,155	7,431,062	462	1844.8	136,106	997,177	70
Trucks7	1076.0	21,745	396,581	27	245.7	35,065	255,020	18
Zeno14	728.4	26,201	6,632,923	421	58.7	17,459	315,719	18

Conclusions

- A new transition based encoding
  - Recent planning formulation SAS+
- Smaller size and faster problem solving
- New encoding can be used to improve other SAT-based planning methods
  - Planning with uncertainty [*Castellini et al. 2003*]
  - Planning with preferences [*Giunchiglia et al. 2007*]
  - Planning with numeric [*Hoffmann et al. 2007*]
  - Temporal planning [*Huang et al. 2009*]

End Part I

- Ancient History: Planning as Satisfiability
  - Planning
  - SAT encoding
  - 3 good ideas:
    - Parallel actions
    - Plan graph pruning
    - Transition based encoding

Part II

- The Future: Markov Logic
  - From random fields to Max-SAT
  - Finite first-order theories
  - 3 good ideas:
    - Lazy inference
    - Query-based instantiation
    - Domain pruning

Slides borrowed freely from Pedro Domingos

Take Away Messages

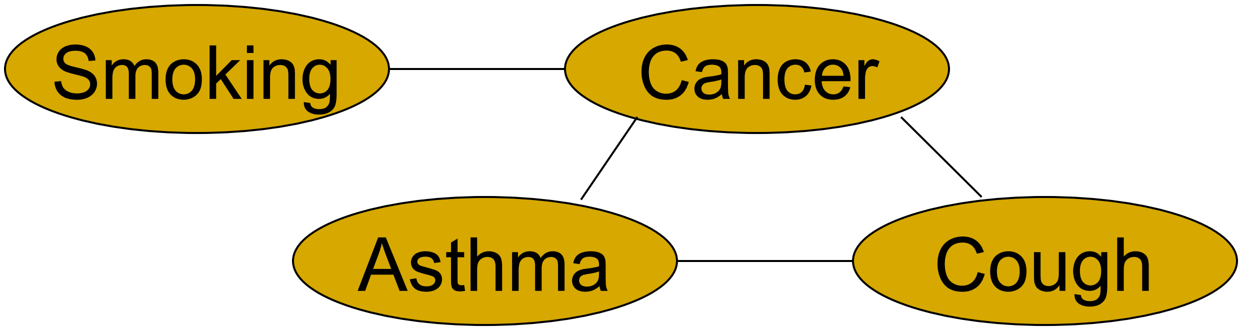
- SAT technology is useful for probabilistic reasoning in graphical models
  - MLE (most like explanation) == MAXSAT
  - Marginal inference == model counting
- Markov Logic is a formalism for graphical models that makes the connection to logic particular clear
- Potential application for SMT

Graphical Models

- Compact (sparse) representation of a joint probability distribution
  - Leverages conditional independencies
  - Graph + associated local numeric constraints
- Bayesian Network
  - Directed graph
  - Conditional probabilities of variable given parents
- Markov Network
  - Undirected graph
  - Un-normalized probabilities (potentials) over cliques

Markov Networks

- **Undirected** graphical models



- Potential functions defined over cliques

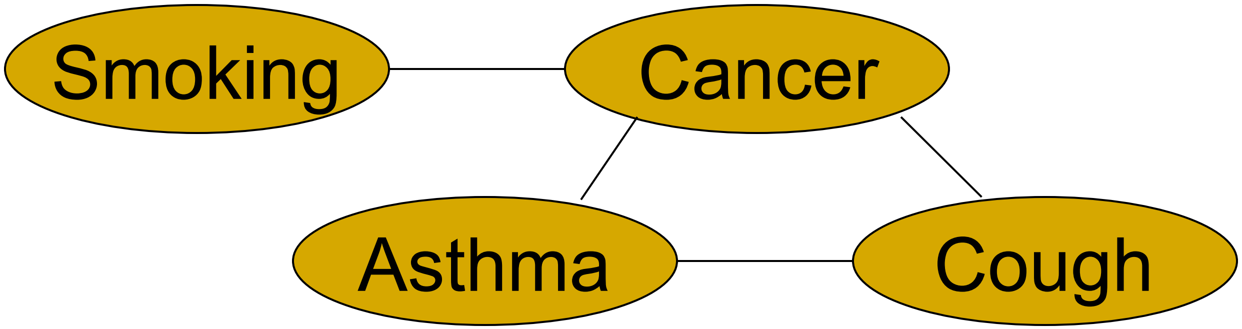
$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	**Cancer**	**Φ(S,C)**
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

Markov Networks

- **Undirected** graphical models



- Log-linear model:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i \underbrace{w_i}_{\text{Weight of Feature } i} \underbrace{f_i(x)}_{\text{Feature } i} \right)$$

$$f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg \text{ Smoking } \vee \text{ Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 1.5$$

Markov Logic: Intuition

- A logical KB is a set of **hard constraints** on the set of possible worlds
- Let's make them **soft constraints**:
  When a world violates a formula,
  It becomes less probable, not impossible
- Give each formula a **weight**
  (Higher weight $\Rightarrow$ Stronger constraint)

$$P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$

Markov Logic: Definition

- A Markov Logic Network (MLN) is a set of pairs (F, w) where
  - F is a formula in first-order logic
  - w is a real number
- Together with a set of constants, it defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w

Example: Friends & Smokers

Smoking causes cancer.

Friends have similar smoking habits.

Example: Friends & Smokers

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$

$$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y) \right)$$

Example: Friends & Smokers

1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Example: Friends & Smokers

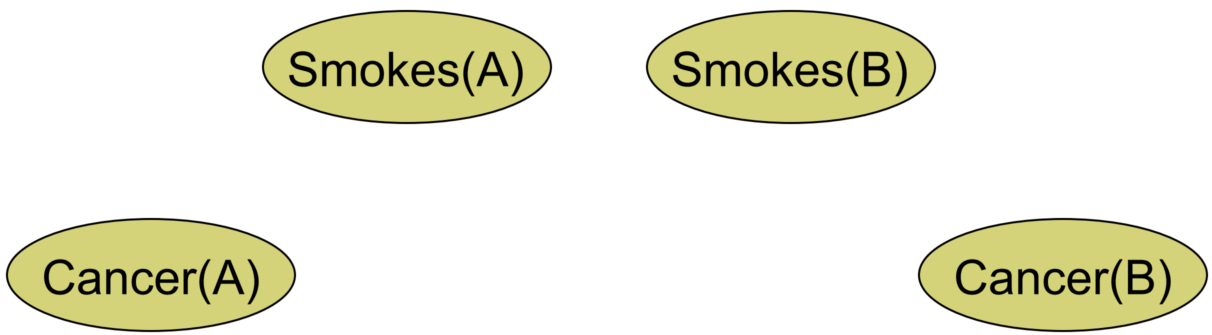
1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)

Example: Friends & Smokers

1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y) \right)$

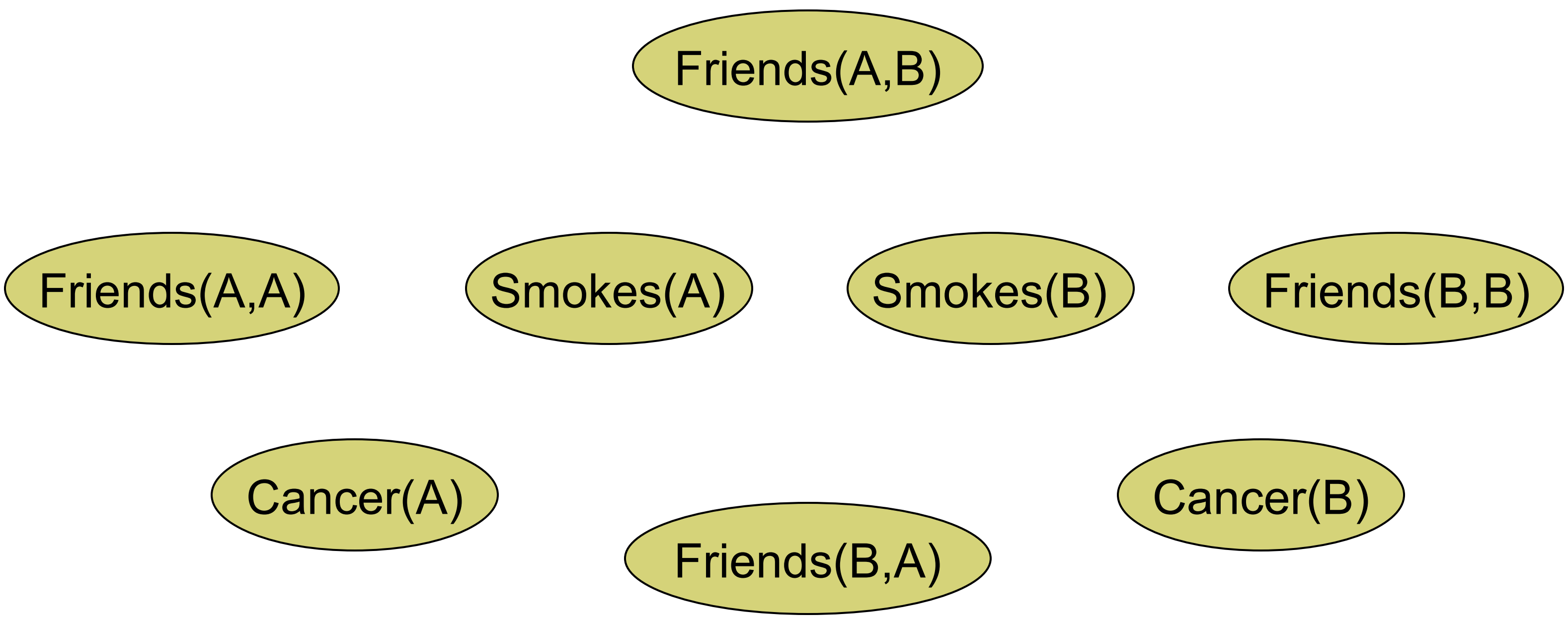
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

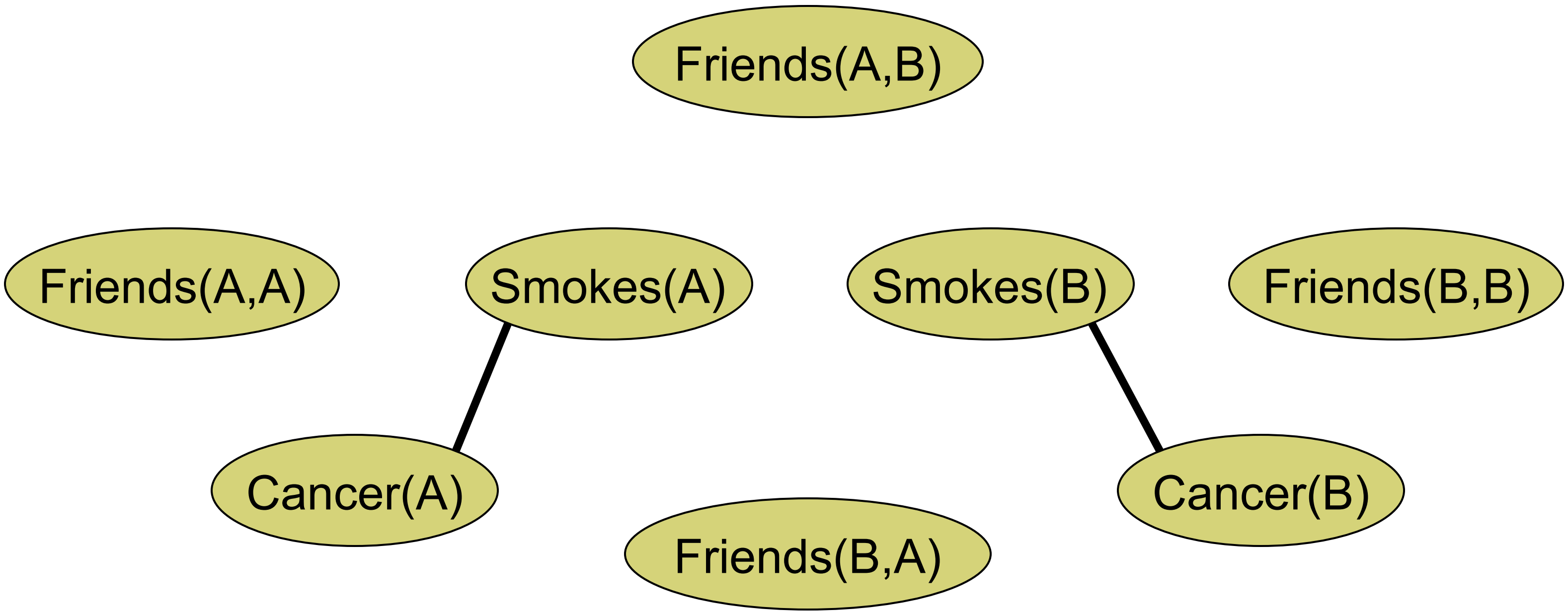
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

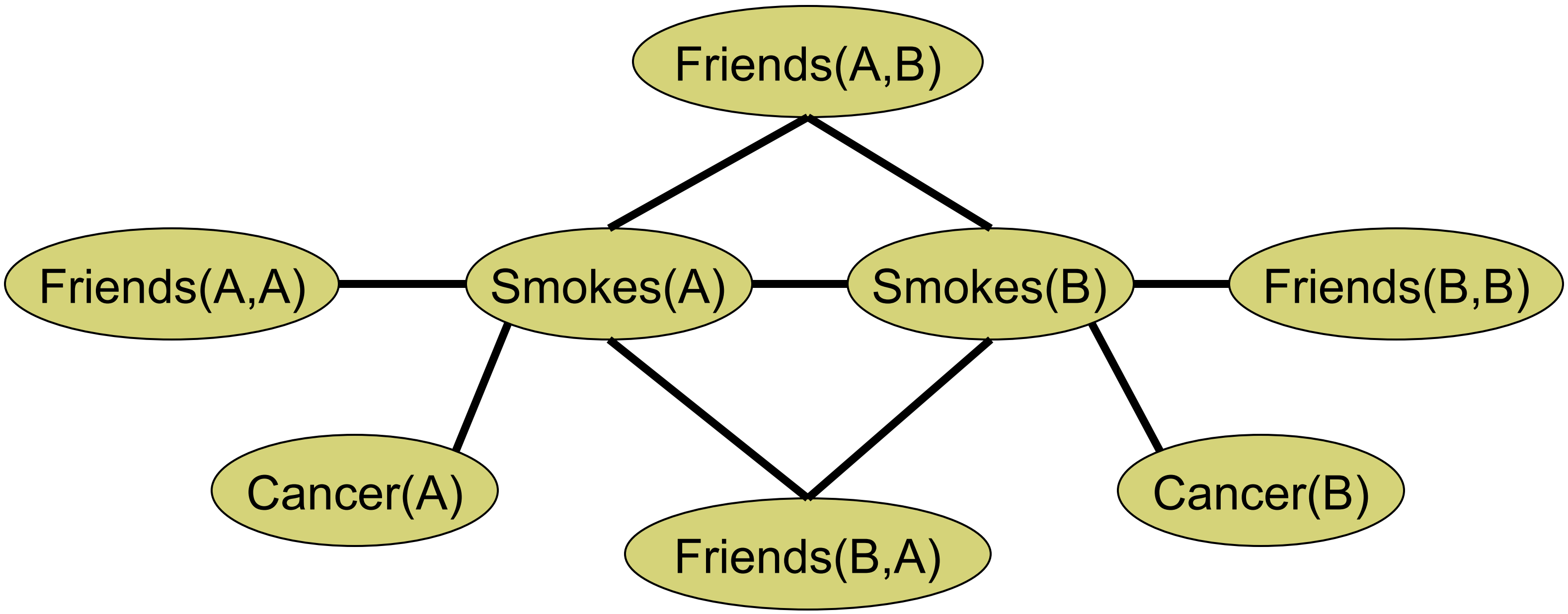
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)



Markov Logic Networks

- MLN is **template** for ground Markov nets
- Probability of a world x:

$$P(x) = \frac{1}{Z}\exp\left(\sum_i \boxed{w_i}\,\boxed{n_i(x)}\right)$$

Weight of formula *i*

No. of true groundings of formula *i* in *x*

- **Typed** variables and constants greatly reduce size of ground Markov net
- Functions, existential quantifiers, etc.
- Infinite and continuous domains

Relation to Statistical Models

- Special cases:
  - Markov networks
  - Markov random fields
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields
- Obtained by making all predicates zero-arity
- Markov logic allows objects to be interdependent (non-i.i.d.)

Relation to First-Order Logic

- Infinite weights $\Rightarrow$ First-order logic
- Satisfiable KB, positive weights $\Rightarrow$ Satisfying assignments = Modes of distribution
- Markov logic allows contradictions between formulas

MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\underbrace{\arg\max_{y}}\ P(y \mid x)$$

Query

Evidence

MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\arg\max_{y} \frac{1}{Z_x} \exp\left(\sum_i w_i n_i(x, y) \right)$$

MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

- This is just the weighted MaxSAT problem
- Use weighted SAT solver (e.g., MaxWalkSAT [Kautz et al., 1997])
- Potentially faster than logical inference (!)

The MaxWalkSAT Algorithm

for *i* ← 1 to *max-tries* **do**
    solution = random truth assignment
    for *j* ← 1 **to** *max-flips* **do**
        if ∑ weights(sat. clauses) > threshold **then**
            return *solution*
        c ← random unsatisfied clause
        with probability *p*
            flip a random variable in *c*
        else
            flip variable in *c* that maximizes
                ∑ weights(sat. clauses)
return failure, best *solution* found

But … Memory Explosion

- **Problem:**
  If there are **n** constants
  and the highest clause arity is **c**,
  the ground network requires $\mathbf{O(n^c)}$ memory

- **Solution:**
  Exploit sparseness; ground clauses lazily
  ⟶ LazySAT algorithm [Singla & Domingos, 2006]
  - Idea: only true literals and unsat clauses need to be kept in memory

Computing Probabilities

- P(Formula|MLN,C) = ?
- MCMC: Sample worlds, check formula holds
- P(Formula1|Formula2,MLN,C) = ?
- If Formula2 = Conjunction of ground atoms
  - First construct min subset of network necessary to answer query (generalization of Knowledge-Based Model Construction)
  - Then apply MCMC (or other)

Ground Network Construction

```
network ← Ø
queue ← query nodes
repeat
    node ← front(queue)
    remove node from queue
    add node to network
    if node not in evidence then
        add neighbors(node) to queue
until queue = Ø
```

Challenge: Hard Constraints

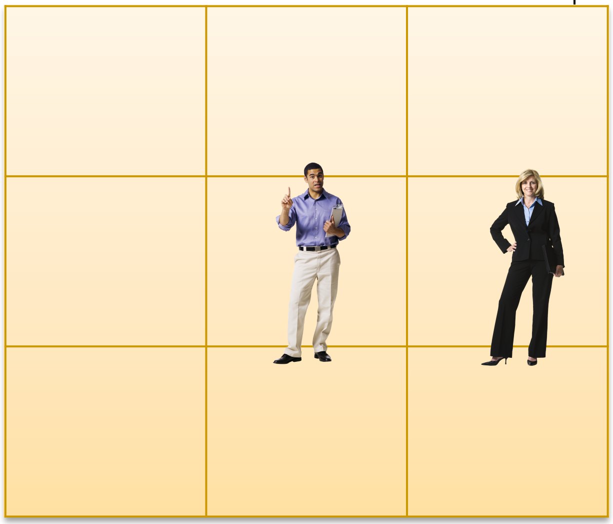
- **Problem:**
  Deterministic dependencies break MCMC
  Near-deterministic ones make it ***very*** slow

- **Solutions:**
  - Combine MCMC and WalkSAT
    $\longrightarrow$ MC-SAT algorithm [Poon & Domingos, 2006]
  - Compilation to arithmetic circuits [Lowd & Domingos 2011]
  - Model counting [Sang & Kautz 2005]

Challenge: Quantifier Degree

- **Problem:**
  Size of instantiated network increases exponentially with quantifier nesting
- **Solution:**
  - Often, most clauses are trivially satisfiable for most entities
  - Preprocess entire theory to infer smaller domains for quantified variables
  - Approach: local consistency (constraint propagation) [Papai, Singla, Kautz 2011]

Example



$\forall$Cell1, Cell2, Agent1, Agent2.

$\quad$ *talk*(Agent1, Agent2) &

$\quad$ *location*(Agent1, Cell1) &

$\quad$ *location*(Agent2, Cell2) $\rightarrow$ *near*(Cell1, Cell2)

$\forall$Cell1, Cell2.

$\quad$ *near*(Cell1, Cell2) $\rightarrow$ Cell1 = Cell2 $\vee$ *adjacent*(Cell1, Cell2)

- 1000 x 1000 grid = 1,000,000 cells
- Previous approach: graphical model is quadratic in number of cells (10^{12} nodes)
- New approach: *linear* in number of cells

Details

- Enforce generalized arc consistency using "hard" constraints
- Efficient implementation using database Join and Project operators
- Reduces total inference time by factor of 2 to 8 on benchmark domains

Domain	Time (in mins)				Ground Tuples (in 1000's)			
	Const. Propagation		Prob. Inference		Const. Propagation		Prob. Inference	
	Stand.	CPI	Stand.	CPI	Stand.	CPI	Stand.	CPI
CTF	0	0.37	1536.6	528.0	0	585.5	2107.8	1308.7
Cora	0	0.07	181.1	26.2	0	153.6	488.2	81.4
Library	0	0.20	286.4	23.0	0	462.7	366.2	45.9

"Constraint Propagation for Efficient Inference in Markov Logic", T. Papai, P. Singla, & H. Kautz, CP 2011.

Alchemy

Open-source software including:

- Full first-order logic syntax
- Generative & discriminative weight learning
- Structure learning
- Weighted satisfiability and MCMC
- Programming language features

alchemy.cs.washington.edu

Capture the Flag Domain

- Rich but controlled domain of interactive activities
  - Very similar to strategic applications
- Rules
  - Two teams, each has a territory
  - A player can be captured when on the opponents' territory
  - A captured player cannot move until freed by a teammate
  - Game ends when a player captures the opponents' flag

Game Video


Map
Satellite
Terrain
Show labels
Traffic
Animate
100 ft
50 m
search the map
Search

Hard Rules for Capturing

H6. A player can only be captured by an enemy.

H7. A player can be captured only when standing on enemy territory.

H9. A player transitions from an uncaptured state to a captured state only via a capture event.

$$\forall a_1, a_2, t : \text{capturing}(a_1, a_2, t) \Rightarrow (\text{enemies}(a_1, a_2) \wedge \text{onEnemyTer}(a_2, t) \wedge \neg\text{onEnemyTer}(a_1, t) \wedge \text{samePlace}(a_1, a_2, t)) \quad \text{(H6, H7)}$$

$$\forall a, t : (\neg\text{isCaptured}(a, t) \wedge \text{isCaptured}(a, t+1)) \Rightarrow (\exists a_1 : \text{capturing}(a_1, a, t)) \quad \text{(H9)}$$

Soft Rules for Capturing

S4. If players a and b are enemies, a is on enemy territory, b is not captured already, and they are snapped to the same location, then *a probably* captures b.

S5. Capture events are generally rare, i.e., there are typically only a few captures within a game.

$$\begin{aligned}\forall a_1, a_2, t : \big[(&\text{enemies}(a_1, a_2) \wedge \text{onEnemyTer}(a_2, t) \wedge \\ &\neg\text{onEnemyTer}(a_1, t) \wedge \text{samePlace}(a_1, a_2, t) \wedge \\ &\neg\text{isCaptured}(a_2, t)) \Rightarrow \text{capturing}(a_1, a_2, t)\big] \cdot w_c\end{aligned} \quad \text{(S4)}$$

$$\forall a, c, t : [\text{capturing}(a, c, t)] \cdot w_{cb} \quad \text{(S5)}$$

Results for Recognizing Captures

	# GPS Readings	# Actual Captures	Baseline	Baseline + States	2-Step ML	Unified ML
Game 1	13,412	2				
Precision			0.006	0.065	1.000	1.000
Recall			1.000	1.000	1.000	1.000
F1			0.012	0.122	1.000	1.000
Game 2	14,400	2				
Precision			0.006	0.011	1.000	1.000
Recall			0.500	0.500	0.833	1.000
F1			0.013	0.022	0.909	1.000
Game 3	3,472	6				
Precision			0.041	0.238	1.000	1.000
Recall			0.833	0.833	0.833	0.833
F1			0.079	0.317	0.909	0.909

Sadilek & Kautz AAAI 2010

End Part II

- The Future: Markov Logic
  - From random fields to Max-SAT
  - Finite first-order theories
  - 3 good ideas:
    - Lazy inference
    - Query-based instantiation
    - Domain pruning