SAT ∩ 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 → 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

mapping → length

plan → interpreter

satisfying model → SAT engine

cnf formula
Clause Schemas

\[ \forall x \in \{A, B, C\} P(x) \]
represents
\[ P(A) \land P(B) \land P(C) \]

\[ \exists x \in \{A, B, C\} P(x) \]
represents
\[ P(A) \lor P(B) \lor P(C) \]
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 $i$

$\forall i \in \text{Times} \quad \forall p \in \text{Planes} \quad \forall a \in \text{Cities} \quad \forall b \in \text{Cities}$

$$\text{fly}(p,a,b,i) \supset (\text{at}(p,a,i)) \land \text{fuel}(p,i))$$
SAT Encoding

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

$\forall i \in \text{Times} \quad \forall p \in \text{Planes} \quad \forall a \in \text{Cities} \quad \forall b \in \text{Cities}$

$$\text{fly}(p,a,b,i) \supset (\text{at}(p,b,i+1)) \land \neg \text{at}(p,a,i+1) \land \neg \text{fuel}(p,i+1)$$
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

\[
\forall i \in \text{Times} \\
\forall p \in \text{Planes} \\
\forall b \in \text{Cities} \\
(\neg \text{at}(p, b, i) \land \text{at}(p, b, i+1)) \supset \exists a \in \text{Cities} . \text{fly}(p, a, b, i)
\]
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:

\[
\forall i \in \text{Times} \\
\forall p \in \text{Planes} \\
\forall a \in \text{Cities} \\
(at(p,a,i) \land \neg at(p,a,i+1)) \supset \exists b \in \text{Cities} . \ fly(p,a,b,i)
\]
Action Mutual Exclusion

- Two actions cannot occur at the same time

\[ \forall i \in \text{Times} \]
\[ \forall p_1, p_2 \in \text{Planes} \]
\[ \forall a, b, c, d \in \text{Cities} \]
\[ \neg \text{fly}(p_1, a, b, i) \lor \neg \text{fly}(p_2, c, d, i) \]

operator: Fly(p,a,b)
precondition: At(p,a), Fueled(p)
effect: At(p,b), \neg At(p,a), \neg Fueled(p)

Constant types: Times, Planes, Cities
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

\[
\forall i \in \text{Times} \\
\forall p_1, p_2 \in \text{Planes} \\
\forall a, b, c, d \in \text{Cities} \\
\neg \text{fly}(p_1, a, b, i) \lor \neg \text{fly}(p_2, c, d, i)
\]
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!
  – Search for a subset of the graph that
    • Supports all the goal propositions
    • Contains no mutually-exclusive propositions
### Initial State

<table>
<thead>
<tr>
<th>facts</th>
<th>actions</th>
<th>facts</th>
<th>actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **action a**: pre \( p \); effect \( \neg p, q \)
- **action b**: pre \( p \); effect \( p \)
- **action c**: pre \( p, q \); effect \( r \)
action a: pre $p$; effect $\neg p$, $q$
action b: pre $p$; effect $p$
action c: pre $p$, $q$; effect $r$
Propagating Mutual Exclusion

action a: pre $p$; effect $\neg p$, $q$

action b: pre $p$; effect $p$

action c: pre $p$, $q$; effect $r$
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 be included in the translation
  – Logically redundant
  – May help or hinder particular SAT solvers
Actions imply preconditions and effects
\[ a_1 \rightarrow p_0 \]
\[ a_1 \rightarrow q_2 \]
\[ a_1 \rightarrow \neg q_2 \]

Facts imply (disjunction of) supporting actions
\[ q_2 \rightarrow a_1 \]
\[ p_2 \rightarrow b_1 \]

Mutual exclusions
\[ \neg a_1 \lor \neg b_1 \]
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+

V(pkg)

in truck

at loc1

at loc2

V(truck)

at loc1

at loc2

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+

<table>
<thead>
<tr>
<th></th>
<th>STRIPS</th>
<th>SAS+</th>
</tr>
</thead>
</table>
| **Definition** | a set of preconditions,  
a set of add effects,  
a set of delete effects | A set of transitions |
| **Example** | (LOAD pkg truck loc1) | pkg:(loc1→truck)  
truck: (loc1→loc1) |
| **Pre:**  | (at truck loc1),  
(at pkg 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

SAT Instance (Part 1):
transitions

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

SAS+ Based New Encoding

Strips Based Encoding

Planning graph

SAT Instance:
Facts and actions

Actions

Transitions

Actions
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

Find matchings

set of actions  set of actions  ...
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:
  \begin{align*}
  x_t &\rightarrow (a_t \land b_t \land c_t) \\
  y_t &\rightarrow (b_t \land c_t \land d_t) \\
  z_t &\rightarrow (a_t \land c_t \land d_t)
  \end{align*}

- Transition implies actions:
  \begin{align*}
  a_t &\rightarrow (x_t \lor z_t) \\
  b_t &\rightarrow (x_t \lor y_t) \\
  c_t &\rightarrow (x_t \lor y_t \lor z_t) \\
  d_t &\rightarrow (y_t \lor z_t)
  \end{align*}

- Action mutual exclusions:
  \begin{align*}
  x_t &\rightarrow \neg y_t; \\
  y_t &\rightarrow \neg z_t; \\
  z_t &\rightarrow \neg x_t
  \end{align*}

These clauses repeat in each time step $t$. 
# Strips v.s. SAS+ Based Encodings

<table>
<thead>
<tr>
<th>Variables</th>
<th>Strips</th>
<th>SAS+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions and Facts</td>
<td></td>
<td>Actions and Transitions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Strips</th>
<th>SAS+</th>
</tr>
</thead>
</table>
| 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]

<table>
<thead>
<tr>
<th>Worst case state space size:</th>
<th>Strips</th>
<th>SAS+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O((2^A)^N)$</td>
<td></td>
<td>$O((2^T)^{NK})$</td>
</tr>
</tbody>
</table>

*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

<table>
<thead>
<tr>
<th>Instances</th>
<th>SatPlan06</th>
<th>New Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec)</td>
<td>#Variables</td>
</tr>
<tr>
<td>Airport40</td>
<td>2239.4</td>
<td>327,515</td>
</tr>
<tr>
<td>Driverslog17</td>
<td>2164.8</td>
<td>61,915</td>
</tr>
<tr>
<td>Freecell4</td>
<td>364.3</td>
<td>17582</td>
</tr>
<tr>
<td>Openstack4</td>
<td>212.1</td>
<td>3,709</td>
</tr>
<tr>
<td>Pipesworld12</td>
<td>3147.3</td>
<td>30,078</td>
</tr>
<tr>
<td>TPP30</td>
<td>3589.7</td>
<td>97,155</td>
</tr>
<tr>
<td>Trucks7</td>
<td>1076.0</td>
<td>21,745</td>
</tr>
<tr>
<td>Zeno14</td>
<td>728.4</td>
<td>26,201</td>
</tr>
</tbody>
</table>
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

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

\[
Z = \sum_x \prod_c \Phi_c(x_c)
\]

<table>
<thead>
<tr>
<th>Smoking</th>
<th>Cancer</th>
<th>(\Phi(S,C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>2.7</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Markov Networks

- **Undirected** graphical models

- Log-linear model:

  \[
  P(x) = \frac{1}{Z} \exp \left( \sum_i w_i f_i(x) \right)
  \]

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

  \[
  w_1 = 1.5
  \]
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

\[ \forall x \ Smokes(x) \Rightarrow Cancer(x) \]
\[ \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \]
Example: Friends & Smokers

<table>
<thead>
<tr>
<th></th>
<th>1.5</th>
<th>$\forall x , Smokes(x) \Rightarrow Cancer(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.1</td>
<td>$\forall x, y , Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$</td>
</tr>
</tbody>
</table>
## Example: Friends & Smokers

<table>
<thead>
<tr>
<th>1.5</th>
<th>$\forall x \ Smokes(x) \Rightarrow Cancer(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$\forall x, y \ Friends(x, y) \Rightarrow (\text{Smokes}(x) \iff \text{Smokes}(y))$</td>
</tr>
</tbody>
</table>

Two constants: **Anna** (A) and **Bob** (B)
Example: Friends & Smokers

Two constants: **Anna** (A) and **Bob** (B)
Example: Friends & Smokers

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

Two constants: Anna (A) and Bob (B)
Example: Friends & Smokers

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

1.5  \( \forall x \ Smokes(x) \Rightarrow Cancer(x) \)
1.1  \( \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \)
Example: Friends & Smokers

1.5 \( \forall x \, \text{Smokes}(x) \Rightarrow \text{Cancer}(x) \)

1.1 \( \forall x, y \, \text{Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y)) \)

Two constants: Anna (A) and Bob (B)
Markov Logic Networks

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

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

- **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

\[
\arg\max_y P(y \mid x)
\]

Query \hspace{3cm} 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

\[ \arg \max_y \sum_i w_i n_i(x, y) \]
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

\begin{algorithm}
    \textbf{for} \( i \leftarrow 1 \) \textbf{to} max-tries \textbf{do}
    \hspace{1em} solution = random truth assignment
    \textbf{for} \( j \leftarrow 1 \) \textbf{to} max-flips \textbf{do}
    \hspace{2em} if \( \sum \) weights(sat. clauses) > threshold \textbf{then}
    \hspace{3em} return solution
    \hspace{2em} \( c \leftarrow \) random unsatisfied clause
    \hspace{2em} with probability \( p \)
    \hspace{3em} flip a random variable in \( c \)
    \textbf{else}
    \hspace{3em} flip variable in \( c \) that maximizes \( \sum \) weights(sat. clauses)
    \textbf{return} failure, best solution found
\end{algorithm}
But ... Memory Explosion

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

- $P(\text{Formula}|\text{MLN}, C) = ?$
- MCMC: Sample worlds, check formula holds
- $P(\text{Formula}_1|\text{Formula}_2, \text{MLN}, C) = ?$
- If $\text{Formula}_2 = \text{Conjunction of ground atoms}$
  - First construct min subset of network necessary to answer query (generalization of Knowledge-Based Model Construction)
  - Then apply MCMC (or other)
Ground Network Construction

\[
\begin{align*}
\text{network} & \leftarrow \emptyset \\
\text{queue} & \leftarrow \text{query nodes} \\
\text{repeat} & \\
& \quad \text{node} \leftarrow \text{front(queue)} \\
& \quad \text{remove node from queue} \\
& \quad \text{add node to network} \\
& \quad \text{if node not in evidence then} \\
& \quad \quad \text{add neighbors(node) to queue} \\
\text{until} & \quad \text{queue} = \emptyset
\end{align*}
\]
Challenge: Hard Constraints

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

- **Solutions:**
  - Combine MCMC and WalkSAT
    → 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

\[ \forall \text{Cell1, Cell2, Agent1, Agent2}. \]
\[ \text{talk}(\text{Agent1, Agent2}) \& \]
\[ \text{location}(\text{Agent1, Cell1}) \& \]
\[ \text{location}(\text{Agent2, Cell2}) \rightarrow \text{near}(\text{Cell1, Cell2}) \]

\[ \forall \text{Cell1, Cell2}. \]
\[ \text{near}(\text{Cell1, Cell2}) \rightarrow \text{Cell1} = \text{Cell2} \lor \text{adjacent}(\text{Cell1, Cell2}) \]

- 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

<table>
<thead>
<tr>
<th>Domain</th>
<th>Time (in mins)</th>
<th>Ground Tuples (in 1000’s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stand.</td>
<td>CPI</td>
</tr>
<tr>
<td>CTF</td>
<td>0</td>
<td>0.37</td>
</tr>
<tr>
<td>Cora</td>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>Library</td>
<td>0</td>
<td>0.20</td>
</tr>
</tbody>
</table>

“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
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.

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

\[
\forall a, t : (\neg \text{isCaptured}(a, t) \land \text{isCaptured}(a, t + 1)) \Rightarrow \\
(\exists a_1 : \text{capturing}(a_1, a, t)) \quad (\text{H9})
\]
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.

\[ \forall a_1, a_2, t : [\text{enemies}(a_1, a_2) \land \text{onEnemyTer}(a_2, t) \land \neg \text{onEnemyTer}(a_1, t) \land \text{samePlace}(a_1, a_2, t) \land \neg \text{isCaptured}(a_2, t) \Rightarrow \text{capturing}(a_1, a_2, t)] \cdot w_c \]  

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

(S4)  

(S5)
Results for Recognizing Captures

<table>
<thead>
<tr>
<th></th>
<th># GPS Readings</th>
<th># Actual Captures</th>
<th>Baseline</th>
<th>Baseline + States</th>
<th>2-Step ML</th>
<th>Unified ML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Game 1</strong></td>
<td>13,412</td>
<td>2</td>
<td>0.006</td>
<td>0.065</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.012</td>
<td>0.122</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td><strong>Game 2</strong></td>
<td>14,400</td>
<td>2</td>
<td>0.006</td>
<td>0.011</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.500</td>
<td>0.500</td>
<td>0.833</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.013</td>
<td>0.022</td>
<td>0.909</td>
<td>1.000</td>
</tr>
<tr>
<td><strong>Game 3</strong></td>
<td>3,472</td>
<td>6</td>
<td>0.041</td>
<td>0.238</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.833</td>
<td>0.833</td>
<td>0.833</td>
<td>0.833</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.079</td>
<td>0.317</td>
<td>0.909</td>
<td>0.909</td>
</tr>
</tbody>
</table>

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