OBDD-based Planning with Real Variables in a Non-Deterministic Environment

Anuj Goel and K. S. Barber

Laboratory for Intelligent Processes and Systems

The University of Texas At Austin

AAAI-99 Student Poster Session

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

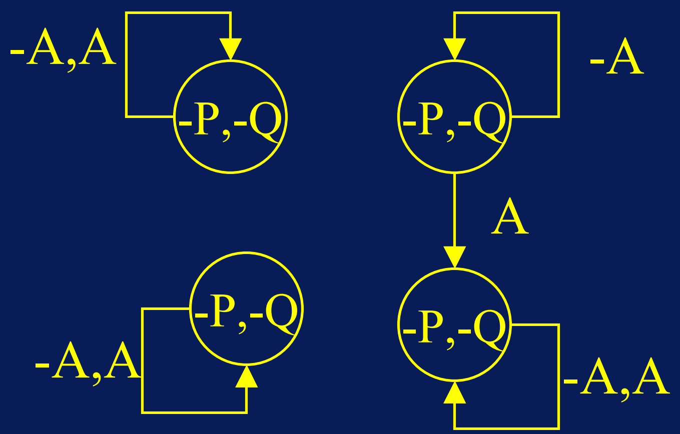
Background

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Action Languages

- In general, action languages represent states (using **fluents**) and transitions (using **actions**)
- Simple example in *C* where A is an **action** and P,Q are **fluents**.

  caused P **if** P **after** P,
  caused -P **if**-P **after** -P,
  caused Q **if** Q **after** Q,
  caused -Q **if** -Q **after** -Q,
  caused P **if** TRUE **after** Q^A.

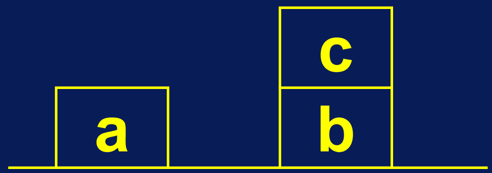


- **STRIPS** -- (Fikes & Nilsson, 1971)
- ***A,B,C*** -- (Gelfond & Lifschitz, 1998)
- **PDDL** -- emerging standard for action description

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Current Process

Assume a blocks world with 3 blocks and portion of an action language description



Action Language

caused on(B,B1) **after** move(B,B1)

Moving a block B onto B1 means B is on B1 at next time step

nonexecutable move(B,B1) **if** on(B2,B) && on(B3,B1)

Moving a block B onto B1 is impossible if either B or B1 have another block on them

Grounding

$$on(a,a)_1 \equiv move(a,a)_0 \wedge \neg on(a,a)_0 \wedge \neg on(b,a)_0 \wedge \neg on(c,a)_0 \wedge \neg on(a,a)_0 \wedge \neg on(b,a)_0 \wedge \neg on(c,a)_0$$

$$on(a,b)_1 \equiv move(a,b)_0 \wedge \neg on(a,a)_0 \wedge \neg on(b,a)_0 \wedge \neg on(c,a)_0 \wedge \neg on(a,b)_0 \wedge \neg on(b,b)_0 \wedge \neg on(c,b)_0$$

$$on(a,c)_1 \equiv move(a,c)_0 \wedge \neg on(a,a)_0 \wedge \neg on(b,a)_0 \wedge \neg on(c,a)_0 \wedge \neg on(a,c)_0 \wedge \neg on(b,c)_0 \wedge \neg on(c,c)_0$$

x 3 x plan length

Pass to SAT Checker

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Satisfiability (SAT) Checkers

- A variety of satisfiability checkers are available for planning problems:
  - **VIS** -- (Brayton et al., 1996)
  - **SMV/NuSMV** -- (Manzo, 1998)
  - **WalkSAT** -- (Selman et al., 1994)

- **Question:** How to apply satisfiability research efficiently in the causal planning domain in order to mitigate state space explosion and improve planning speed?

Query Language Support

- Given a possible set of initial states and actions --

  Query languages formulate a set of queries concerning the system's future state

  - P,Q,R (Gelfond & Lifschitz, 1998) - Query languages for the *A,B,C* set of action languages
  - CTL (Computational Tree Logic) - Widely used standard in satisfiability research and logic synthesis
  - Various implementation specific query languages developed by individual researchers

Problems with State-of-the-Art

- **State Space Explosion**
  - Grounded representation size dependent on plan length, number of actions, number of fluents and number of possible parameters
  - Instantiation of all plan times results in heavy performance penalty for replanning
- **Query Languages**
  - Query languages vary between action languages; leading to confusion
- **Satisfiability Checking**
  - Usage of CNF for state encoding produces slow satisfiability checking for large problems

Proposed Improvements

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

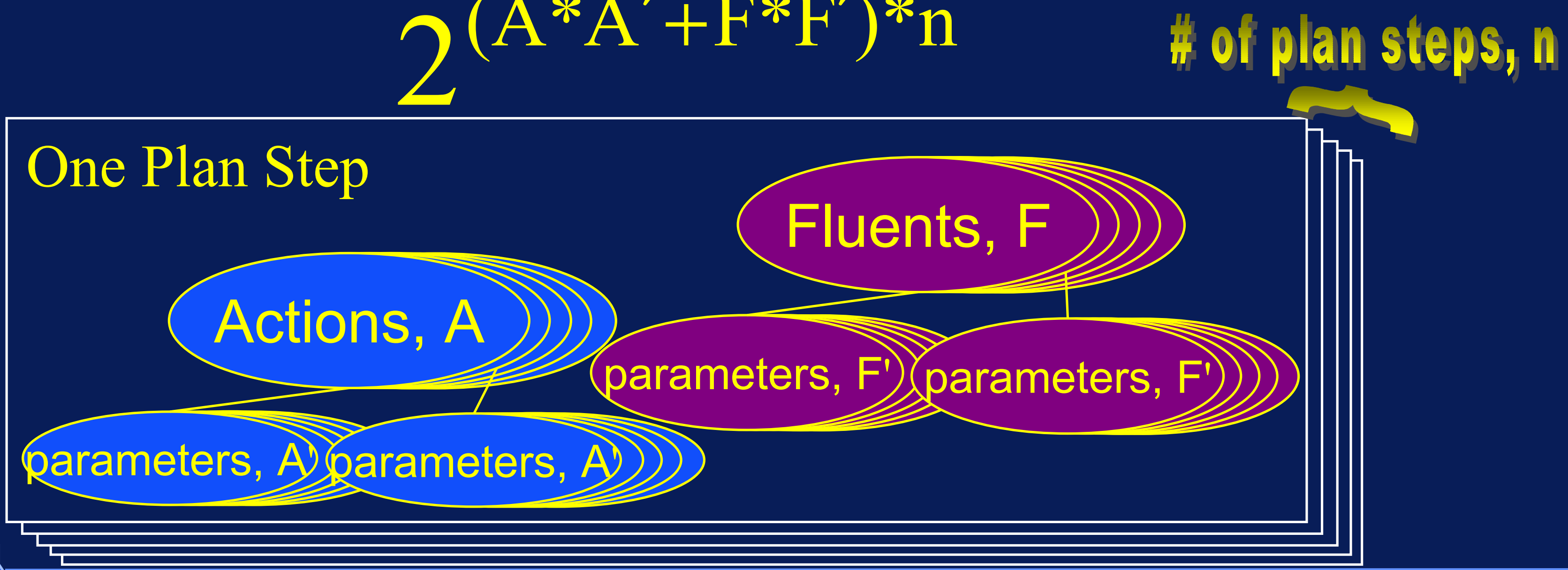
Proposed Theoretical Improvements

- **State Space Reduction**
  - Innovative use of new encodings facilitated by new satisfiability checkers
- **Query Language Expressiveness**
  - Use of standards from other fields (e.g. CTL)
- **Encoding for Satisfiability Checking**
  - BDD (Binary Decision Diagram)
  - Efficient compact representation of states provided by certain satisfiability tools

State Space Reduction (I)

- **Expected size:**
  - **A = # of actions at any given time**
  - **A'= Average # of possible parameters on any action A**
  - **F = # of fluent variables**
  - **F'= Average # of parameters on any action F**
  - **n = # of time steps in plan**

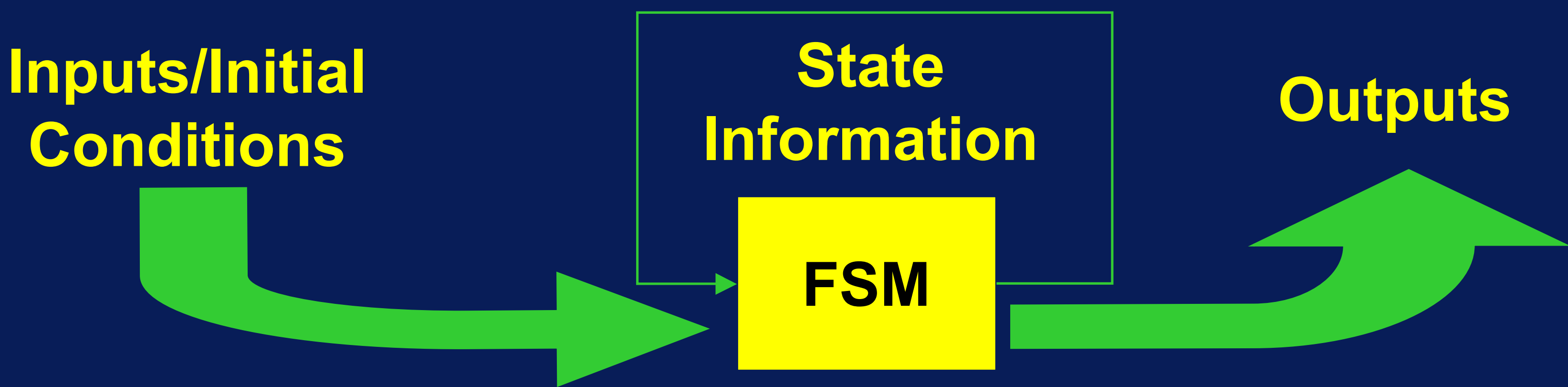
$$2^{(A*A'+F*F')*n}$$



©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

State Space Reduction (II)

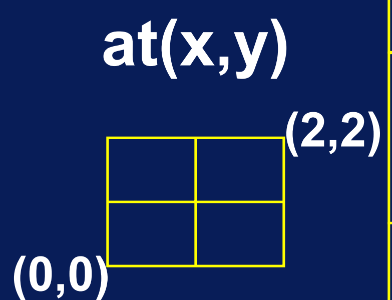
- **Approach: State-based Encodings**
  - Reduce state space by using a Finite State Machine and calculating available next states.
  - Dynamic environment = lots of replanning, current methods ground representation of unreached states
- **Impact:**
  - Reduces memory usage by only encoding current and next state
  - Grounded state space size not related to plan length; results in a reduction by a factor of 2^n



©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

State Space Reduction(III)

- **Most current tools:**
  - requires explicitly instantiation of each numerical parameter
  - force relative boolean representations to describe absolute values.
- **Approach: Parameterized Encoding**
  - does not require explicit instantiation
  - allows direct representation of numerical values
- **Impact:**
  - State space reduction of $2^{A'}$



Encoding	Ground State	Comments
Explicit	at(2,0), at(2,1), at(2,2) at(1,0), at(1,1), at(1,2) at(0,0), at(0,1), at(0,2)	A total of 9 variables are needed.
Boolean	above(bottom), near(left), etc.	Absolute positioning is lost and all position is relative
Parameter	at(int x, int y)	Preserves positioning and requires one variable; increases computation reqs.

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

State Space Reduction (IV)

- **Intelligent branching - (Giungchiglia,et al. 1998)**
  - Many current SAT planners do not differentiate between fluents and actions when searching the state space.
- **Approach:**
  - Note: Changes in fluents are the result of actions.
  - Any fluents whose values can be deterministically chosen by action assignments can be pruned.
- **Impact:**
  - Reduction of $2^{(F*F')}$ where **F** is a deterministically derived fluent value and **F'** is the average # of possible parameters.

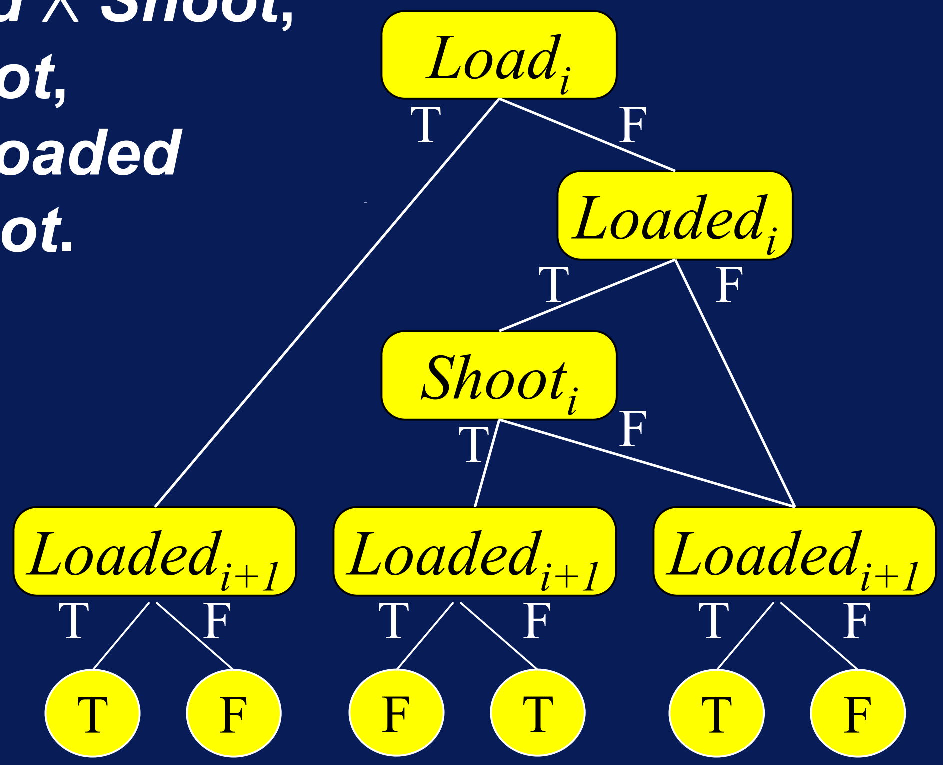
Query Language Expressiveness

- **Approach:**
  - Support for standard CTL syntax provides access to standard query representation without sacrificing expressiveness.
  - CTL Syntax:
    - AF(x) - x will be always eventually true (always finally)
    - AG(x) - x is always true (always globally)
    - EF(x) - it is possible for x to be true (eventually finally)
    - EG(x) - it is possible for x to eventually always be true (eventually globally)
- **Impact:**
  - Provides a common language-independent representation accepted by many existing tools

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

BDD - Binary Decision Diagram (I)

interial *Loaded*, ¬*Loaded*, *Alive*, ¬*Alive*,
caused *Loaded* after *Load*,
caused ¬*Alive* after *Loaded* ∧ *Shoot*,
caused ¬*Loaded* after *Shoot*,
nonexecutable *Shoot* if ¬*Loaded*
nonexecutable *Load* ∧ *Shoot*.



©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

BDD - Binary Decision Diagram (II)

- **Approach:**
  - BDDs supported by a variety of SAT checkers
  - Provide an efficient and compact encoding of state
- **Impact:**
  - Reduction in memory usage for representing grounded states
  - Faster query language checking from SAT checkers
  - Faster plan solutions from usage of SAT checkers

Current Implementation

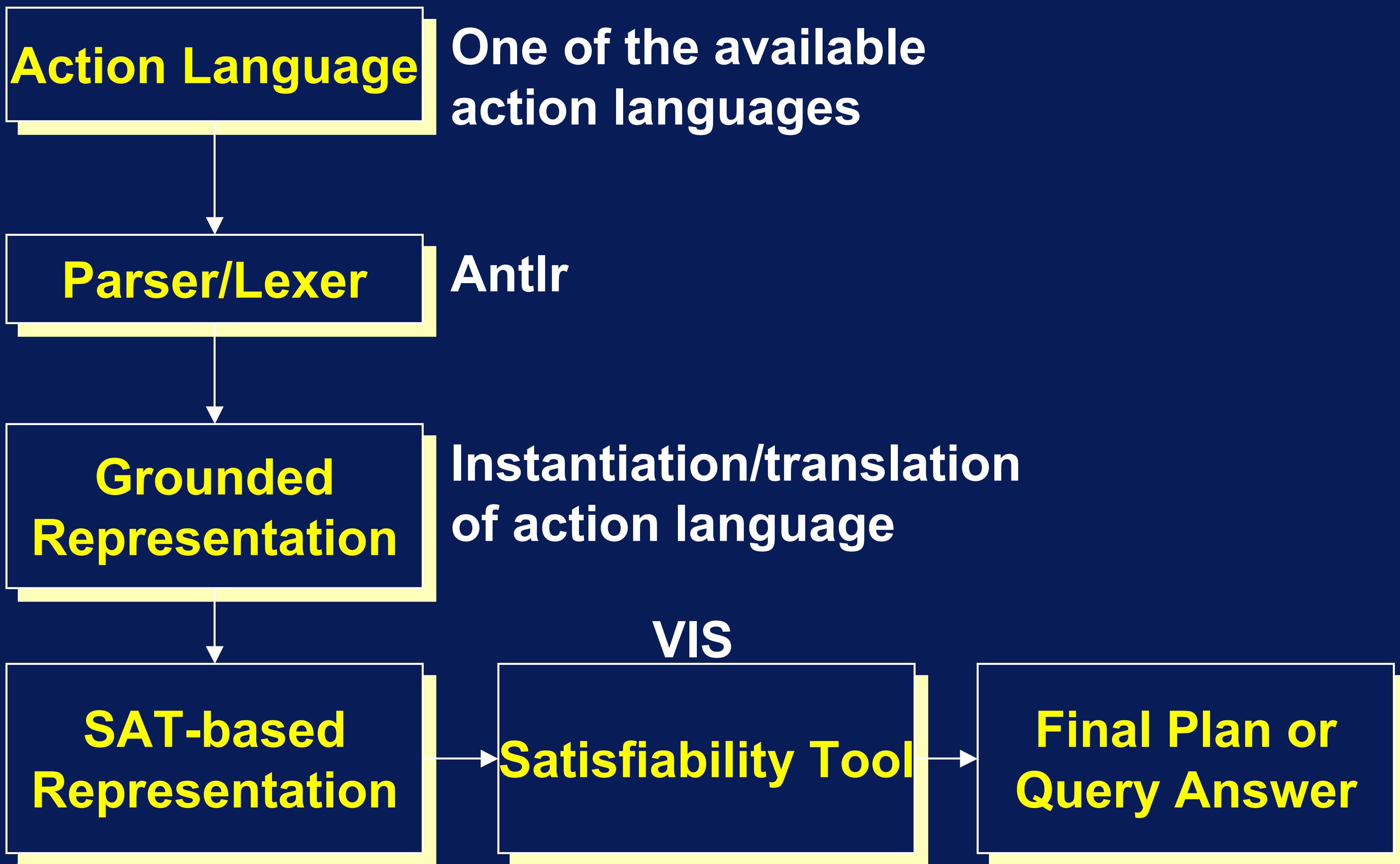
©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Research Leveraging Existing Tools

- VIS → A satisfiability checker and verification tool
- C → An advanced action language representation
- BLIF-MV → A logic file format that can be accepted by VIS.
- Antlr → A lex/yacc type parsing tool

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Architecture



©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Current State of Research

- Causal Parser implementation is complete
  - grounding and generation of SAT-based representation is being explored.
- Numerical value usage within a SAT checker is being explored.
- Speed/size testing against other planners remains to be done.

©1999 The Laboratory for Intelligent Processes and Systems, The University of Texas at Austin

Conclusions

- SAT tools have been shown to perform efficiently when used for planning tasks.
- Improvements are possible to:
  - Enhance the language expressiveness
  - Improve query utilization through standards usage
- Usage of these techniques may reduce memory requirements and increase speed to plan solution