Planning as Model Checking
Given a domain (states and actions), an initial and goal state, the planning problem is the problem to find a plan of actions that leads from the initial state to the goal state. The planning problem is the problem to find a plan of actions that leads from the initial state to the goal state. The planning problem is the problem to find a plan of actions that leads from the initial state to the goal state.

Planning Problem:

- **Goal State** ("Blue on Green", "Green on Red", "Red on Table")
- **Initial State** ("Red on Table", "Blue on Table", "Green on Blue")
- **Domain = States (block positions) + Actions (moves)

Plan: "Move Green on Red", then "Move Blue on Green"
A Hard Problem!

5 locations, 3 piles, 3 trucks, 100 containers \( \rightarrow \) 10277 states!
The Planner automatically generates the plan of actions.

The Controller observes the current state and executes actions.

System

Controller

Plan

Domain

Initial

Goal

Actions

Observations
The "Classical" Planning Problem

Restrictive assumption: no uncertainty, complete knowledge

Deterministic Domain: actions from one state lead to a single state
Full Observability: complete knowledge about the state of the system
Goal: a set of final desired states

The Controller

The Planner

System

Actions

Observations

Goal

Initial

Plan
Partial Observability

(top view)

(side view)
For instance: "Keep the system infinitely often through a given state."

Extended Goals
A realistico esempio (II)
State of the art is still an open problem!

Practical automated planning with non-determinism, partial observability and extended goals is still an open problem!

1. Classical Planning: no applicable partial observability and for extended goals.
2. (Limited) extensions to classical planning: no practical partial observability, and for extended goals.
3. Expressive frameworks (e.g., deductive): no automatic planning

State of the art w.r.t. planning in non-deterministic domains, under
Structure of the talk

1. Planning as Model Checking in Non-Deterministic Domains
   (Alessandro Cimatti)

2. Planning under Partial Observability
   (Piergiorgio Bertoli)

3. Planning under Null Observability
   (Marco Roveri)

4. Planning for Extended Goals
   (Marco Pistore)

5. Conclusions and ... Dreams
Structure of the talk

1. Planning as Model Checking in Non-Deterministic Domains
   - Planning via Symbolic Model Checking (Alessandro Cimatti)
   - Planning for Extended Goals (Marco Pistore)
   - Planning under Partial Observability (Piergiorgio Bertoli)
   - Planning under Null Observability (Marco Roveri)

2. Conclusions and Dreams
Model Checking is a formal verification technique if a system, represented as a Finite State Machine (FSM), satisfies certain properties, represented as temporal logic formulae. Model checking algorithms: exhaustive traversal of the model.

Model Checker is software tool for system verification:

- Model checking algorithms: exhaustive traversal of the model
- Satisfies certain properties, represented as temporal logic formulae
- A system, represented as Finite State Machine (FSM)

Remark: high industrial applicability!
Reducing a Planning problem to a Model Checking problem:

- Domain represented as a Finite State Machine
- Actions as inputs to the FSM
- Observation as Machine Outputs
- Action effects as transitions
- Temporal formula

Planning as Model Checking:

1. Plan as counterexample traces
2. Planning as exploration of the FSM
3. Observations as Machine Outputs
4. Action effects as transitions
5. Actions as inputs to the FSM
6. Domain represented as a Finite State Machine

Model Checker

Finite-state transition system

Yes!

no!

Counterexample
Symbolic Model Checking

Key issue: extremely large state spaces!

Symbolic Model Checking:

Represent sets of states of the FSM symbolically, by means of boolean operations (e.g.: union, intersection) as transforms over states of the FSM symbolically.

Boolean formulas:

A BDD represents the assignments satisfying (and falsifying) a boolean formula as ordered binary decision diagrams (BDDs).

Operations over sets of states (e.g.: union, intersection) as transformations over BDDs.

\[ X \land \neg X \]

\[ X \lor X \]

\[ \neg X \]

\[ X \]
Symbolic State-Space Exploration

The state-space is explored by iterating preimages (i.e., states) from which a certain set is reachable, until a fix point is reached. Implemented as transformations over BDDs.
NuSMV: A Symbolic Model Checker

NuSMV is available at http://sra.itc.it/tools/nusmv/

- copyleft: Improvements made available for distribution.
- tree availability for commercial usage.

Public License:

Soon to be released for Open Source development under Mozilla

Industrial controllers:

Used (by IRST) in technology transfer projects in design of

- used for teaching courses in several universities;
- basis for several Ph.D. theses;

Over 150 downloads since July 1999:

A joint IRST-CMU development;

NuSMV: A Symbolic Model Checker developed at IRST
Planning via Symbolic Model Checking

Inherits the basic technology of symbolic model checking;
Extends symbolic model checking;
Planning is a harder problem than model checking;
Planning under full observability;
Planning under partial observability (conformant planning);
Planning under null observability (conformant planning);
Conditional planning under full observability;
Conditional planning under partial observability;
Conditional planning under null observability.

MBP is available at http://sra.itc.it/tools/mbp/
MBP: a Model Based Planner
based on the NuSMV model checker.

MBP is a Model Based Planner
Planning for temporally extended goals.
Planning under partial observability;
Planning under full observability (conformant planning);
Planner under null observability (conformant planning);
Functionalities:
- Conditional planning under full observability;
- Conditional planning under partial observability;
- Conditional planning under null observability;
- Planning for temporally extended goals.

The Planning via Symbolic Model Checking Paradigm:

Plannnig via Symbolic Model Checking

Functionalities:
- Planning for temporally extended goals.
- Planning under partial observability;
- Planning under full observability (conformant planning);
- Planning under null observability (conformant planning);
- Conditional planning under full observability;
- Conditional planning under partial observability;
- Conditional planning under null observability;
- Planning for temporally extended goals.

MBP is available at http://sra.itc.it/tools/mbp/
Structure of the talk

1. Planning as Model Checking in Non-Deterministic Domains
   (Alessandro Cimatti)
2. Planning under Null Observability (Marco Roveri)
3. Planning under Partial Observability (Piergiorgio Bertoli)
4. Planning for Extended Goals (Marco Pistore)
5. Conclusions and Dreams
A simple example: The Blind Robot Problem (Michie '74)

Domain:
- Actions: GO(pos), PICK, LETGO
- GOOUT requires key at door
- GOOUT selects randomly an object among the possible ones.

Initially:
- Robot position unknown.
- Robot hand empty.
- Domain:

Goal:
- Red object located at out.
- Objects initially located at door are known to be red.
- Either one of objects A or B has a key.
Conformant Plan for the Blind Robot Problem

**Key Problems:**
- Uncertainty in the initial condition.
- Non-deterministic action effects.
- Unlabeledness of indistinguishable states.

**Our Approach:**
- Represent a Belief State as a BDD.

**Key States:**
- Belief States:
- Non-deterministic action effects.

**Problems:**
- Our Approach: Represent a Belief State as a BDD.
Conformant Planning: Algorithms, Intuitions

Basic search step: expansion of a belief state

Forward:

Termination: Bs, or search space exhausted.

Search style: breadth-first, depth-first, best-first, A*.

Role of Symbolic Model Checking Techniques:

Primitives:
- Symbolic expansion of belief states (extends symbolic expansion)
- Visited belief states represented as BDDs, stored in a hash table.

Forward Search

Backward Search

Symbolic expansion of belief states (extends symbolic expansion)

Visited belief states represented as BDDs, stored in a hash table.

Forward: α, β, γ, δ are applicable in I and result in Bs1, Bs2, Bs3.

Backward: α, β, γ, δ are applicable in I and result in Bs1, Bs2, Bs3.

Basic search step: expansion of a belief state
Conformant Planning: Results

Two classes of algorithms:

- **Fully-symbolic search**: additional BDD variables to encode the plan associated to a belief state.
- **Heuristic-symbolic search**: the plan is stored in the hash table of visited belief states.

Experimental results show that the algorithms are effective in finding optimal (i.e., of minimal length) solutions when the selection function is admissible (A* style search) solutions. Algorithms allow for different selection functions thus exploiting different search strategies (e.g., depth-first, breadth-first, best-first).

When the selection function is admissible (A* style search), solutions found are optimal. With a failure otherwise, a solution for problems admitting a conformant solution terminates with the algorithms guaranteed to terminate.

Conformant Planning: Results
Experimental Evaluation: Planning Domains

- CPT heuristic, non-symbolic plannner (best competitor)
- MBP heuristic-symoblic search
- MBP fully-symbolic search
The reset sequence problem: find a sequence of inputs that takes a system into a known state from any initial state.

**Goal:** the set of “certain” (singleton) belief states.

- e.g. the Pentium processor at power-up.

**Our approach:** forward algorithms with modified termination tests.

**Results:** our technique compares positively with state-of-the-art (BDD based) specialized algorithms.
Structure of the talk

1. Planning as Model Checking in Non-Deterministic Domains (Alessandro Cimatti)
2. Planning under Partial Observability (Piergiorgio Bertoli)
3. Planning under Null Observability (Marco Roveri)
4. Planning for Extended Goals (Marco Pistore)
5. Conclusions and Dreams

Planning via Symbolic Model Checking (Alessandro Cimatti)
Thus, we reason about sets of states: belief states (BS).

- Because actions may have unpredictable results...

- Because the initial situation is uncertain...

Under Po, in general states cannot be exactly determined.
An action transforms a BS into a BS

Observing some feature may split a BS

Planning: Key concepts
Search space and conditional plan

GoEast; if WallN then {GoSouth; GoWest} else GoWest
Results

Goals:
- Propositional formulas on final states.

Plans:
- Strong acyclic (non-iterative) conditional.

Planning algorithms:
- Guaranteed to terminate for planning problems admitting such a plan.
- Failure otherwise.
- A variety of search styles.

Implementation of the planning algorithms by BDD-based symbolic model checking techniques.

Experimental results show that the algorithms work in practice.
Experimental Results
Structure of the talk

1. Planning as Model Checking in Non-Deterministic Domains: (Alessandro Cimatti)
2. Planning under Partial Observability: (Piergiorgio Bertoli)
3. Planning under Null Observability: (Marco Roveri)
4. Planning for Extended Goals: (Marco Pistore)
5. Conclusions and Dreams
Extended Goals: some examples

"Deliver n objects in n given rooms"
Extended Goals: some examples

"Keep delivering objects to given rooms as they are generated"

Producer rooms
Extended Goals: some examples

"Keep TRYING to deliver objects to given rooms as they are generated"

"Kid doors"
Extended Goals are CTL formulas

Intuitions:

\( \text{AXp} \)

Abbreviations:

\( (\top \land b) \exists \equiv b \exists \)  \( (\top \land b) \forall \equiv b \forall \)

\( (b \land \bot) \exists \equiv b \exists \)  \( (b \land \bot) \forall \equiv b \forall \)

\( (b \land b) \exists \land (b \land b) \forall \land (b \land b) \exists \land (b \land b) \forall \)

Extended Goals are CTL formulas
Controller: while
end while

\begin{align*}
\text{s} & := \text{SENSE\_CURRENT\_STATE()} \\
\text{EXEC}(a); \\
\text{a} & := \text{GET\_ACTION}(s, \text{plan}) \\
\text{do} & \text{while } s \in \text{STATE\_FOR\_PLAN}(\text{plan}) \\
\text{s} & := \text{SENSE\_CURRENT\_STATE()} \\
\end{align*}

Controller:

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>„enter room“</td>
<td>„door open“</td>
</tr>
<tr>
<td>„open door“</td>
<td>„door closed“</td>
</tr>
<tr>
<td>„move to door“</td>
<td>„object grasped“</td>
</tr>
<tr>
<td>„object not grasped“</td>
<td>„gasp object“</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Planning Algorithm

Goal as formulas in Temporal Logic, e.g.,
\[ 5 < M^2 : N \]

 Means "Find a plan that guarantees that \( p \) is maintained and that leaves the possibility to reach \( q \)."

\[ \text{Progress}(\text{EF} q) = q \lor \text{EF} q \]
\[ \text{Progress}(\text{AG} p) = p \land \text{AG} x \]

Control Automaton

Domain Automaton

\[ \text{AG} p \]
\[ \text{AG} p \]

Controller Automaton

Goal as formulas in Temporal Logic, e.g.,
\[ b \text{ EF } d \]
\[ \text{AG} \]
Automatic generation of conditional, iterative, and history dependent plans for any possible extended goal expressed as a CTL formula. The planning algorithm is correct and complete: it generates plans that are guaranteed to satisfy the specifications for all possible non-deterministic behaviour of the system. If no plan exists, it is guaranteed to terminate with failure.

Experimental results show that, in spite of its generality, the algorithm works in practice.
<table>
<thead>
<tr>
<th>k</th>
<th>p = 1</th>
<th>p = 2</th>
<th>p = 3</th>
<th>p = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Parameters:**
- p = # of producer rooms
- k = # of kid-doors

**Goal:**

\[
\forall x \left( \bigvee_{i=1}^{d} \mathcal{A} \right)
\]
1. Planning as Model Checking in Non-Deterministic Domains:
   - Alessandro Cimatti
2. Planning under Null Observability (Marco Roveri)
3. Planning under Partial Observability (Piergiorgio Bertoli)
4. Planning for Extended Goals (Marco Pistore)
5. Conclusions and ... Dreams

Structure of the talk


References:

IJCAI - International Joint Conferences on Artificial Intelligence,
Future work

Future work

Planning and execution

Planning with noisy sensors

Planning with probabilities

Planning for extended goals under partial observability

...
The Dream of Automatic Synthesis

Theorem proving approach:
extract a program from a proof that the specification is satisfiable

Automata theoretic approach:
transformed into an automaton

Planning as model checking approach:
from specification (goal) + (planning) domain a plan is generated

From specification (goal) + (planning) domain a plan is generated
Conclusions

Planning as Model Checking: well-founded, general, and practical

Possible future impacts: e.g., a new approach to synthesis, ...

Current impact within the planning community

Planning as Model Checking: well-founded, general, and practical
Thanks to...

Moshhe Vardi (RICE University)

Marco Danelée (PhD at University of Rome)

Enrico Giunchiglia (University of Genoa)

Fausto Giunchiglia (IRST and University of Trento)