Angelica Hierarchical Planning: Optimal and Online Algorithms

Bhaskara Marthi
MIT/Willow Garage
bhaskara@csail.mit.edu

Stuart Russell
UC Berkeley
russell@cs.berkeley.edu

Jason Wolfe
UC Berkeley
jawolfe@cs.berkeley.edu
High-Level Actions (HLAs)

- Here, a **high-level action (HLA)** =
  a set of allowed **immediate refinements**:
  - each is a **sequence** of actions
  - may have associated preconditions
- Almost all actions we think about are high-level
  - Plan a trip
  - Vacuum the house
  - Go to work

\[
\text{[Go(work)]}
\]

\[
\text{[GetIn(car), Drive(work), GetOut(car)]} \quad \text{if } \neg \text{Raining} \quad \text{[Walk(stop), Bus(work)]}
\]
Abstract Lookahead

• $k$-step lookahead $\gg$ 1-step lookahead
  • e.g., chess
Abstract Lookahead

• $k$-step lookahead $>>$ 1-step lookahead
  • e.g., chess
• $k$-step lookahead no use if steps too small
  • e.g., first $k$ turns in TSP of Australia
Abstract Lookahead

- $k$-step lookahead $\gg$ 1-step lookahead
  - e.g., chess
- $k$-step lookahead no use if steps too small
  - e.g., first $k$ turns in TSP of Australia
Abstract Lookahead

- $k$-step lookahead $>>$ 1-step lookahead
  - e.g., chess
- $k$-step lookahead no use if steps too small
  - e.g., first $k$ turns in TSP of Australia
  - this is one small part of a human life, $\approx 20,000,000,000,000$ primitive actions
Abstract Lookahead

• *k*-step lookahead >> 1-step lookahead
  • e.g., chess
• *k*-step lookahead no use if steps too small
  • e.g., first *k* turns in TSP of Australia
  • this is one small part of a human life, ≈ 20,000,000,000,000 primitive actions
• Abstract plans with HLAs are shorter
Abstract Lookahead

- *k*-step lookahead >> 1-step lookahead
  - e.g., chess
- *k*-step lookahead no use if steps too small
  - e.g., first *k* turns in TSP of Australia
  - this is one small part of a human life, \( \approx 20,000,000,000,000 \) primitive actions
- Abstract plans with HLAs are shorter
  - Much shorter plans => exponential savings
Abstract Lookahead

- $k$-step lookahead $>>$ 1-step lookahead
  - e.g., chess
- $k$-step lookahead no use if steps too small
  - e.g., first $k$ turns in TSP of Australia
  - this is one small part of a human life, $\approx 20,000,000,000,000$ primitive actions
- Abstract plans with HLAs are shorter
  - Much shorter plans $\Rightarrow$ exponential savings
  - Can look ahead much further

looks like a good start
Abstract Lookahead

- \( k \)-step lookahead \( \gg \) 1-step lookahead
  - e.g., chess
- \( k \)-step lookahead no use if steps too small
  - e.g., first \( k \) turns in TSP of Australia
  - this is one small part of a human life,
    \( \approx 20,000,000,000,000 \) primitive actions
- **Abstract** plans with HLAs are shorter
  - Much shorter plans \( \Rightarrow \) exponential savings
  - Can look ahead much further
- Requires **models** for HLAs
  - i.e., transition and cost fns

*looks like a good start*
Abstract Lookahead

- *k*-step lookahead >> 1-step lookahead
  - e.g., chess
- *k*-step lookahead no use if steps too small
  - e.g., first *k* turns in TSP of Australia
  - this is one small part of a human life,
    \( \approx 20,000,000,000,000 \) primitive actions

- **Abstract** plans with HLAs are shorter
  - Much shorter plans => exponential savings
  - Can look ahead much further
- Requires **models** for HLAs
  - i.e., transition and cost fns
  - No suitable models in literature

*looks like a good start*
Abstract Lookahead

- k-step lookahead >> 1-step lookahead
  - e.g., chess
- k-step lookahead no use if steps too small
  - e.g., first k turns in TSP of Australia
  - this is one small part of a human life, \( \approx 20,000,000,000,000 \) primitive actions

- **Abstract** plans with HLAs are shorter
  - Much shorter plans => exponential savings
  - Can look ahead much further
- Requires **models** for HLAs
  - i.e., transition and cost fns
  - No suitable models in literature
  - We extend our **angelic semantics**
Angelic Semantics for HLAs [MRW ’07]

- Models HLAs in deterministic domains
Angelic Semantics for HLAs [MRW ’07]

- Models HLAs in deterministic domains
- Central idea is reachable set of an HLA from some state
Angelic Semantics for HLAs [MRW '07]

- Models HLAs in deterministic domains
- Central idea is reachable set of an HLA from some state
  - When extended to sequences of actions, ...
Angelic Semantics for HLAs [MRW '07]

• Models HLAs in deterministic domains
• Central idea is **reachable set** of an HLA from some state
  • When extended to sequences of actions, ...

[Diagram of state space and transitions]
Angelic Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is reachable set of an HLA from some state
  • When extended to sequences of actions, ...
Angelic Semantics for HLAs [MRW ’07]

- Models HLAs in deterministic domains
- Central idea is reachable set of an HLA from some state
  - When extended to sequences of actions, ...
Angelic Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is **reachable set** of an HLA from some state
  • When extended to sequences of actions, ...
Angelic Semantics for HLAs [MRW ’07]

- Models HLAs in deterministic domains
- Central idea is *reachable set* of an HLA from some state
  - When extended to sequences of actions, ...
  - … allows proving that a plan can or cannot possibly reach the goal
Angelic Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is reachable set of an HLA from some state
  • When extended to sequences of actions, ...
  • … allows proving that a plan can or cannot possibly reach the goal
Angelic Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is reachable set of an HLA from some state
  • When extended to sequences of actions, ...
  • … allows proving that a plan can or cannot possibly reach the goal

\[ [h_1, h_2] \text{ is a solution} \]
Angelics Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is reachable set of an HLA from some state
  • When extended to sequences of actions, ...
  • … allows proving that a plan can or cannot possibly reach the goal
• May seem related to nondeterminism …
Angelics Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is reachable set of an HLA from some state
  • When extended to sequences of actions, ...
  • … allows proving that a plan can or cannot possibly reach the goal
• May seem related to nondeterminism ...
  • but uncertainty is angelic: resolved by the agent, not an adversary

\[[h_1, h_2] \text{ is a solution} \]

State space

\[[h_1, h_2] \]

G

\[s_0\]
Angelic Semantics for HLAs [MRW ’07]

• Models HLAs in deterministic domains
• Central idea is *reachable set* of an HLA from some state
  • When extended to sequences of actions, ...
  • … allows *proving* that a plan can or cannot possibly reach the goal
• May seem related to nondeterminism ...
  • but uncertainty is *angelic*: resolved by the agent, not an adversary
Angelic Semantics cont.

- Approximate descriptions provide **lower & upper bounds** on reachable sets
  - Descriptions are **true**: follow logically from hierarchy
Angelic Semantics cont.

• Approximate descriptions provide lower & upper bounds on reachable sets
  • Descriptions are true: follow logically from hierarchy

• Sound & complete planning algorithm uses descriptions to
  • Commit to provably successful abstract plans: Downward Refinement Property (DRP) automatically satisfied
    • potentially exponential speedup
  • Prune provably unsuccessful abstract plans (USP satisfied)
Contributions

• Extend angelic semantics with action costs

• Developed novel algorithms that do lookahead with HLAs
  • Angelic Hierarchical A* (AHA*)
  • Angelic Hierarchical Learning Real-Time A* (AHLRTA*)

• Both require three inputs:
  • planning problem
  • action hierarchy (set of HLAs)
  • approximate models for HLAs
Deterministic Planning Problems

• Here, a planning problem =
Deterministic Planning Problems

• Here, a planning problem =
  • State space $S$
Deterministic Planning Problems

- Here, a planning problem =
  - State space $S$
  - Initial state $s_0$, terminal set $G$
Deterministic Planning Problems

- Here, a planning problem =
  - State space $S$
  - Initial state $s_0$, terminal set $G$
  - Primitive action set
Deterministic Planning Problems

Here, a planning problem =

- State space $S$
- Initial state $s_0$, terminal set $G$
- Primitive action set
- Transition function: $S \times A \rightarrow S$
- Cost function: $S \times A \rightarrow \mathbb{R} \cup \{\infty\}$
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: Warehouse World Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row

- All actions have cost 1

- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps
Running Example: Warehouse World Domain

- Elaborated Blocks World with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: *Warehouse World* Domain

- Elaborated *Blocks World* with discrete spatial constraints
  - Gripper must stay in bounds
  - Can’t pass through blocks
  - Can only turn at top row
- All actions have cost 1
- Goal: have C on T4
  - Can’t just move directly
  - Final plan has 22 steps

Running Example: Warehouse World HLAs

L  D  GetR  U  Turn  D  PutL
Running Example: *Warehouse World* HLAs
Running Example: *Warehouse World* HLAs

```
NavT(2,3)
  /   \
Nav(2,3)  NavT(2,3)
  \   /           \   /
    L  D          Nav(3,3)  Nav(2,3)
          \       /           \
GetR    U  Turn  D  PutL
```
Running Example: *Warehouse World* HLAs

```
NavT(2,3)  NavT(2,3)
  Nav(2,3)  Nav(3,3)  Nav(2,3)
    L      D  GetR  U  Turn  D  PutL
```

Move(C,A)
Running Example: *Warehouse World* HLAs

```
Act

Move(C,A)

NavT(2,3)

Nav(2,3)

L  D  GetR

NavT(2,3)

Nav(3,3)

U  Turn

Nav(2,3)

D  PutL
```

...
Running Example: *Warehouse World* HLAs

- Plans of interest are *primitive refinements* of special HLA Act

[Act]
Running Example: *Warehouse World* HLAs

- Plans of interest are *primitive refinements* of special HLA `Act`
- Each HLA has a set of *immediate refinements* into action sequences

```
[Act]
...
iff at G

[Move(B,C), Act]
```
Running Example: *Warehouse World* HLAs

- Plans of interest are **primitive refinements** of special HLA `Act`

- Each HLA has a set of **immediate refinements** into action sequences

```
[Act]
  ...
  iff at G
  ...
  [Move(B,C), Act]
  ...
  ...
  [NavT(left of B), GetR, NavT(left of target), PutR]
```
Running Example: *Warehouse World* HLAs

- Plans of interest are **primitive refinements** of special HLA Act
- Each HLA has a set of **immediate refinements** into action sequences

```
[NavT(left of B), GetR, NavT(left of target), PutR]
```

```
[Move(B,C), Act]
```

```
Act
```

```
iff at G
```

```
... ... ...
```
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)

```
Act
Move(C,A) ----> Action
Move(A,C) ----> Action
```
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: **refine** a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
**Abstract Lookahead Trees (ALTs)**

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
Abstract Lookahead Trees (ALTs)

- ALTs generalize lookahead trees for flat algs (e.g., A*)
  - Represent a set of potential plans
  - Basic operation: refine a plan (replace with all refs. at some HLA)
  - Nodes have optimistic & pessimistic valuations
Modeling HLAs

- An HLA is fully characterized by planning problem + hierarchy
Modeling HLAs

• An HLA is fully characterized by planning problem + hierarchy
  • But without abstraction, lose benefits of hierarchy
Modeling HLAs

• An HLA is fully characterized by planning problem + hierarchy
  • But without abstraction, lose benefits of hierarchy

• Extension of idea from “Angelic Semantics for HLAs” [MRW ‘07]:
  • Valuation of HLA $h$ from state $s$:
    • For each $s'$, min cost of any primitive refinement of $h$ that takes $s$ to $s'$

NavT(0, 1)
Modeling HLAs

• An HLA is fully characterized by planning problem + hierarchy
  • But without abstraction, lose benefits of hierarchy

• Extension of idea from “Angellic Semantics for HLAs” [MRW ‘07]:
  • Valuation of HLA \( h \) from state \( s \):
    • For each \( s' \), \textit{min cost} of any primitive refinement of \( h \) that takes \( s \) to \( s' \)
  • Exact description of \( h \) = valuation of \( h \) from each \( s \)
Modeling HLAs

• An HLA is fully characterized by planning problem + hierarchy
  • But without abstraction, lose benefits of hierarchy

• Extension of idea from “Angelic Semantics for HLAs” [MRW ‘07]:
  • Valuation of HLA \( h \) from state \( s \):
    • For each \( s' \), \textit{min cost} of any primitive refinement of \( h \) that takes \( s \) to \( s' \)
  • Exact description of \( h = \) valuation of \( h \) from each \( s \)
  • But this description has no compact, efficient representation in general
Optimistic and Pessimistic Valuations

• Instead, use \textit{approximate} valuations
Optimistic and Pessimistic Valuations

• Instead, use approximate valuations

• We choose a simple form: reachable set + cost bound on set
Optimistic and Pessimistic Valuations

- Instead, use approximate valuations
- We choose a simple form: reachable set + cost bound on set
- Optimistic valuations never overestimate best achievable cost
Optimistic and Pessimistic Valuations

• Instead, use approximate valuations
• We choose a simple form: reachable set + cost bound on set
• Optimistic valuations never overestimate best achievable cost
• Pessimistic valuations never underestimate best achievable cost
Representing Descriptions: NCSTRIPS
Representing Descriptions: NCSTRIIPS

- Descriptions specify propositions *(possibly)* added/deleted by HLA
Representing Descriptions: NCSTRIPS

- Descriptions specify propositions (possibly) added/deleted by HLA

\[
\text{NavT}(x_t, y_t) \quad \text{(Pre: At}(x_s, y_s))
\]
Representing Descriptions: NCSTrips

- Descriptions specify propositions (possibly) added/deleted by HLA
  - Also include a cost bound

**NavT(x_t,y_t)**

**Opt:**

\[-At(x_s,y_s), +At(x_t,y_t), \simFaceR, \simFaceR\]

\[\text{cost} \geq |x_s - x_t| + |y_s - y_t|\]
Representing Descriptions: NCSTRIPS

- Descriptions specify propositions (possibly) added/deleted by HLA
  - Also include a cost bound
  - Can condition on features of initial state

\[\text{NavT}(x_t, y_t)\]

**Opt:**
- \(-\text{At}(x_s, y_s), +\text{At}(x_t, y_t), \sim\text{FaceR}, \sim\text{FaceR}\)
- \(\text{cost} \geq |x_s - x_t| + |y_s - y_t|\)

**Pess:**
- IF \(\text{Free}(x_t, y_t) \land \forall x \text{Free}(x, y_{\text{max}})\) :
  - \(-\text{At}(x_s, y_s), +\text{At}(x_t, y_t), \sim\text{FaceR}, \sim\text{FaceR}\)
  - \(\text{cost} \leq |x_s - x_t| + 2y_{\text{max}} - y_t - y_s + 1\)
Representing Descriptions: NCSTRLIPS

- Descriptions specify propositions (possibly) added/deleted by HLA
  - Also include a cost bound
  - Can condition on features of initial state

<table>
<thead>
<tr>
<th>NavT(xₜ,yₜ)</th>
<th>(Pre: At(xₛ,yₛ))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Opt:</strong></td>
<td>-At(xₛ,yₛ), +At(xₜ,yₜ), ¬FaceR, ¬FaceR</td>
</tr>
<tr>
<td></td>
<td>cost ≥</td>
</tr>
</tbody>
</table>

| **Pess:**  | IF Free(xₜ,yₜ) ∧ ∀x Free(x,yₘₜ) : |
|            | -At(xₛ,yₛ), +At(xₜ,yₜ), ¬FaceR, ¬FaceR |
|            | cost ≤ |xₛ - xₜ| + 2yₘₜ - yₜ - yₛ + 1 |
| **ELSE:**  | nil |

**NavT(xₜ,yₜ)**

![NavT Diagram](image)

**Opt:**

- At(xₛ,yₛ), +At(xₜ,yₜ), ¬FaceR, ¬FaceR
  - cost ≥ |xₛ - xₜ| + |yₛ - yₜ|

**Pess:**

IF Free(xₜ,yₜ) ∧ ∀x Free(x,yₘₜ) :

- At(xₛ,yₛ), +At(xₜ,yₜ), ¬FaceR, ¬FaceR
  - cost ≤ |xₛ - xₜ| + 2yₘₜ - yₜ - yₛ + 1

**ELSE:**

nil
Representing Descriptions: NCSTRIPS

• Descriptions specify propositions (possibly) added/deleted by HLA
  • Also include a cost bound
  • Can condition on features of initial state

• An simple algorithm progresses a valuation (DNF + #)
  through an NCSTRIPS description to produce next valuation

<table>
<thead>
<tr>
<th>NavT(x_t,y_t)</th>
<th>(Pre: At(x_s,y_s))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opt: -At(x_s,y_s), +At(x_t,y_t), ÷FaceR, ÷FaceR</td>
<td></td>
</tr>
<tr>
<td>cost ≥</td>
<td>x_s - x_t</td>
</tr>
</tbody>
</table>

| Pess: IF Free(x_t,y_t) ∧ ∀x Free(x,y_max) : |
| -At(x_s,y_s), +At(x_t,y_t), ÷FaceR, ÷FaceR |
| cost ≤ |x_s - x_t| + 2y_max - y_t - y_s + 1 |

| ELSE: nil |
Angelic Hierarchical A* (AHA*)

• Construct an ALT with the single plan \([\text{Act}]\)
• Loop
  • Select a plan with minimal optimistic cost to \(G\)
  • If primitive, return it
  • Otherwise, refine one of its HLAs
    • Prune dominated refinements
AHA*: Intuitive Picture

highest-level primitive
AHA*: Intuitive Picture

highest-level

primitive
AHA*: Intuitive Picture

highest-level
primitive
AHA*: Intuitive Picture

highest-level primitive

Act

G

S₀
AHA*: Intuitive Picture

highest-level primitive
AHA*: Intuitive Picture

highest-level primitive
AHA*: Intuitive Picture

highest-level primitive
AHA*: Intuitive Picture

highest-level primitive
AHA*: Intuitive Picture

highest-level
primitive
Analysis of AHA*

- AHA* is **hierarchically optimal** (HO)
  - Optimistic valuation → **admissible heuristic**
  - Pruning never rules out all HO plans
- Better descriptions lead to lower runtime
  - optimistic → directed search
  - pessimistic → pruning (refine HO plans w/o backtracking)
- Reduces to A* given “flat” hierarchy: Act → [Prim, Act]

<table>
<thead>
<tr>
<th>Solution Length</th>
<th>A*</th>
<th>AHA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>4.7</td>
</tr>
<tr>
<td>25</td>
<td>40</td>
<td>11</td>
</tr>
<tr>
<td>37</td>
<td>550</td>
<td>30</td>
</tr>
<tr>
<td>44</td>
<td>&gt; 10000</td>
<td>68</td>
</tr>
</tbody>
</table>

*run times in seconds on five warehouse world instances of increasing solution length*
Online Search

• Situated agents must cope with passage of time
  • offline planning rarely feasible
  • common alternative: real-time search

• Korf’s Learning Real-Time A* (LRTA*):
  • Combines limited lookahead + learning
  • Always reaches goal, converges to optimal

• Angelic Hierarchical LRTA* (AHLRTA*)
  • Performs hierarchical lookahead
  • Shares LRTA*’s guarantees
  • Reduces to LRTA* given “flat” hierarchy
Online Results

1 AHLRTA* refinement \approx 5 LRTA* refinements
Summary

Model-based hierarchical planning is theoretically interesting, shows promising empirical performance