Lecture slides for

*Automated Planning: Theory and Practice*

# Chapter 10
# Control Rules in Planning

Dana S. Nau

University of Maryland

5:01 PM April 4, 2012

# Motivation

- Often, planning can be done much more efficiently if we have domain-specific information
- Example:
  - classical planning is EXPSPACE-complete
  - block-stacking can be done in time $O(n^3)$

- But we don't want to have to write a new domain-specific planning system for each problem!

- *Domain-configurable* planning algorithm
  - Domain-independent search engine (usually a forward state-space search)
  - Input includes domain-specific information that allows us to avoid a brute-force search
    - » Prevent the planner from visiting unpromising states

# Motivation (Continued)

- If we're at some state $s$ in a state space, sometimes a domain-specific test can tell us that
  - $s$ doesn't lead to a solution, or
  - for any solution below $s$, there's a better solution along some other path
- In such cases we can to prune $s$ immediately

```
Abstract-search(u)
    if Terminal(u) then return(u)
    u ← Refine(u)          ;;  refinement step
    B ← Branch(u)          ;;  branching step
    B′ ← Prune(B)          ;;  pruning step
    if B′ = ∅ then return(failure)
    nondeterministically choose v ∈ B′
    return(Abstract-search(v))
end
```

- Rather than writing the domain-dependent test as low-level computer code, we'd prefer to talk directly about the planning domain
- One approach:
  - Write logical formulas giving conditions that states must satisfy; prune states that don't satisfy the formulas
- Presentation similar to the chapter, but not identical
  - Based partly on TLPlan [Bacchus & Kabanza 2000]

# Quick Review of First Order Logic

- First Order Logic (FOL):
  - constant symbols, function symbols, predicate symbols
  - logical connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$), quantifiers ($\forall$, $\exists$), punctuation
  - Syntax for formulas and sentences
    - $on(A,B) \wedge on(B,C)$
    - $\exists x\ on(x,A)$
    - $\forall x\ (ontable(x) \Rightarrow clear(x))$
- First Order Theory $T$:
  - "Logical" axioms and inference rules – encode logical reasoning in general
  - Additional "nonlogical" axioms – talk about a particular domain
  - Theorems: produced by applying the axioms and rules of inference
- Model: set of objects, functions, relations that the symbols refer to
  - For our purposes, a model is some state of the world $s$
  - In order for $s$ to be a model, all theorems of $T$ must be true in $s$
  - $s \models on(A,B)$ read "$s$ satisfies $on(A,B)$" or "$s$ entails $on(A,B)$"
    - » means that $on(A,B)$ is true in the state $s$

# Linear Temporal Logic

- Modal logic: FOL plus *modal operators*

  to express concepts that would be difficult to express within FOL
- Linear Temporal Logic (LTL):
  - Purpose: to express a limited notion of time
    - An infinite sequence $\langle 0, 1, 2, \ldots \rangle$ of time instants
    - An infinite sequence $M = \langle s_0, s_1, \ldots \rangle$ of states of the world
  - Modal operators to refer to the states in which formulas are true:

| | | | | |
|---|---|---|---|---|
| $\bigcirc f$ | - | *next* $f$ | - | $f$ holds in the next state, e.g., $\bigcirc\, on(A,B)$ |
| $\Diamond f$ | - | *eventually* $f$ | - | $f$ either holds now or in some future state |
| $\Box f$ | - | *always* $f$ | - | $f$ holds now and in all future states |
| $f_1 \cup f_2$ | - | $f_1$ *until* $f_2$ | - | $f_2$ either holds now or in some future state, and $f_1$ holds until then |

  - Propositional constant symbols TRUE and FALSE

# Linear Temporal Logic (continued)

- Quantifiers cause problems with computability
  - Suppose $f(x)$ is true for infinitely many values of $x$
  - Problem evaluating truth of $\forall x\ f(x)$ and $\exists x\ f(x)$

- Bounded quantifiers
  - Let $g(x)$ be such that $\{x : g(x)\}$ is finite and easily computed

    $\forall[x{:}g(x)]\, f(x)$
    - means $\forall x\ (g(x) \Rightarrow f(x))$
    - expands into $f(x_1) \wedge f(x_2) \wedge \ldots \wedge f(x_n)$

    $\exists[x{:}g(x)]\, f(x)$
    - means $\exists x\ (g(x) \wedge f(x))$
    - expands into $f(x_1) \vee f(x_2) \vee \ldots \vee f(x_n)$

# Models for LTL

- A model is a triple $(M, s_i, v)$
  - $M = \langle s_0, s_1, \ldots \rangle$ is a sequence of states
  - $s_i$ is the $i$'th state in $M$,
  - $v$ is a *variable assignment* function
    - a substitution that maps all variables into constants

- To say that $v(f)$ is true in $s_i$, write $(M,s_i,v) \models f$

- Always require that

  $(M, s_i, v) \models$ TRUE

  $(M, s_i, v) \models \neg$FALSE

- For planning, need to augment LTL to refer to goal states
  - Include a GOAL operator such that GOAL($f$) means $f$ is true in every goal state
  - $((M,s_i,V),g) \models \text{GOAL}(f)$ iff $(M,s_i,V) \models f$ for every $s_i \in g$

# Examples

- Suppose $M = \langle s_0, s_1, \ldots \rangle$

$(M,s_0,v) \models \bigcirc\bigcirc\, on(A,B)$ means $A$ is on $B$ in $s_2$

- Abbreviations:

$(M,s_0) \models \bigcirc\bigcirc\, on(A,B)$ no free variables, so $v$ is irrelevant:

$M \models \bigcirc\bigcirc\, on(A,B)$ if we omit the state, it defaults to $s_0$

- Equivalently,

$(M,s_2,v) \models on(A,B)$ same meaning with no modal operators

$s_2 \models on(A,B)$ same thing in ordinary FOL

- $M \models \Box\neg holding(C)$
  - in every state in $M$, we aren't holding $C$
- $M \models \Box(on(B, C) \Rightarrow (on(B, C) \cup on\ (A, B)))$
  - whenever we enter a state in which $B$ is on $C$, $B$ remains on $C$ until $A$ is on $B$.

# TLPlan

Procedure TLPlan $(s, f, g, \pi)$

```
if f = FALSE then return failure
if s satisfies g then return π
f⁺ ← Progress (f, s)
if f⁺ = FALSE then return failure
A ← {actions applicable to s}
if A is empty then return failure
nondeterministically choose a ∈ A
s⁺ ← γ(s,a)
return TLPlan (s⁺, f⁺, g, π.a)
```

- Basic idea: forward search, using LTL for pruning tests
- Let $s_0$ be the initial state, and $f_0$ be the initial LTL control formula
- Current recursive call includes current state $s$, and current control formula $f$
- Let $P$ be the path that TLPlan followed to get to $s$
  - The proposed model $M$ is $P$ plus some (not yet determined) states after $s$
- If $f$ evaluates to FALSE in $s$, no $M$ that starts with $P$ can satisfy $f_0$ => *backtrack*
- Otherwise, consider the applicable actions, to see if one of them can produce an acceptable "next state" for $M$
  - Compute a formula $f^+$ that must be true in the next state
    - » $f^+$ is called the **progression** of $f$ through $s$
  - If $f^+$ = FALSE, then there are no acceptable successors of $s$ => backtrack
  - Otherwise, produce $s^+$ by applying an action to $s$, and call TLPlan recursively

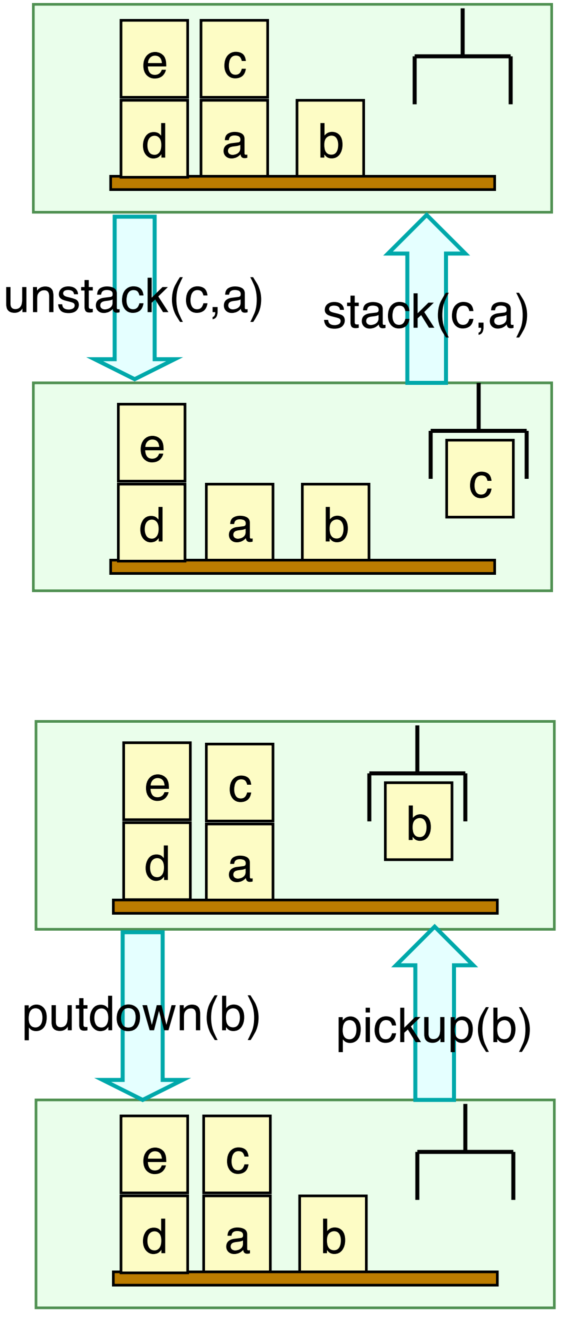
# Classical Operators

unstack($x,y$)
- Precond: on($x,y$), clear($x$), handempty
- Effects: ¬on($x,y$), ¬clear($x$), ¬handempty, holding($x$), clear($y$)

stack($x,y$)
- Precond: holding($x$), clear($y$)
- Effects: ¬holding($x$), ¬clear($y$), on($x,y$), clear($x$), handempty

pickup($x$)
- Precond: ontable($x$), clear($x$), handempty
- Effects: ¬ontable($x$), ¬clear($x$), ¬handempty, holding($x$)

putdown($x$)
- Precond: holding($x$)
- Effects: ¬holding($x$), ontable($x$), clear($x$), handempty



# Supporting Axioms

- Want to define conditions under which a stack of blocks will never need to be moved
- If $x$ is the top of a stack of blocks, then we want *goodtower*($x$) to hold if
  - $x$ doesn't need to be anywhere else
  - None of the blocks below $x$ need to be anywhere else
- Axioms to support this:
  - $goodtower(x) \Leftrightarrow clear(\mathrm{x}) \wedge \neg\, \mathrm{GOAL}(holding(x)) \wedge goodtowerbelow(x)$
  - $goodtowerbelow(x) \Leftrightarrow$

$$[ontable(x) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(x,y))]]$$
$$\vee\ \exists[y{:}on(x,y)]\ \{\neg\mathrm{GOAL}(ontable(x)) \wedge \neg\mathrm{GOAL}(holding(y))$$
$$\wedge\ \neg\mathrm{GOAL}(clear(y)) \wedge \forall[z{:}\mathrm{GOAL}(on(x,z))]\ (z = y)$$
$$\wedge\ \forall[z{:}\mathrm{GOAL}(on(z,y))]\ (z = x) \wedge goodtowerbelow(y)\}$$

  - $badtower(x) \Leftrightarrow clear(x) \wedge \neg goodtower(x)$

# Blocks World Example (continued)

Three different control formulas:

(1) Every goodtower must always remain a goodtower:

$$\Box\Big(\forall[x{:}clear(x)]\ goodtower(x) \Rightarrow \bigcirc(clear(x) \vee \exists[y{:}on(y,x)]\ goodtower(y))\Big)$$

(2) Like (1), but also says never to put anything onto a badtower:

$$\Box\Big(\forall[x{:}clear(x)]\ goodtower(x) \Rightarrow \bigcirc(clear(x) \vee \exists[y{:}on(y,x)]\ goodtower(y)) \wedge badtower(x) \Rightarrow \bigcirc(\neg\exists[y{:}on(y,x)]\ )\Big)$$

(3) Like (2), but also says never to pick up a block from the table unless you can put it onto a goodtower:

$$\Box\Big(\forall[x{:}clear(x)]\ goodtower(x) \Rightarrow \bigcirc(clear(x) \vee \exists[y{:}on(y,x)]\ goodtower(y)) \wedge badtower(x) \Rightarrow \bigcirc(\neg\exists[y{:}on(y,x)]\ ) \wedge (ontable(x) \wedge \exists[y{:}\mathrm{GOAL}(on(x,y))]\ \neg goodtower(y)) \Rightarrow \bigcirc(\neg holding(x))\Big)$$

# Outline of How TLPlan Works

- Recall that TLPLan's input includes a current state $s$, and a control formula $f$ written in LTL
  - How can TLPLan determine whether there exists a sequence of states $M$ beginning with $s$, such that $M$ satisfies $f$?
- We can compute a formula $f^+$ such that for every sequence $M = \langle s, s^+, s^{++}, \ldots \rangle$,
  - $M$ satisfies $f$ iff $M^+ = \langle s^+, s^{++}, \ldots \rangle$ satisfies $f^+$
- $f^+$ is called the **progression** of $f$ through $s$
- If $f^+$ = FALSE then there is no $M^+$ that satisfies $f^+$
  - Thus there's no $M$ that begins with $s$ and satisfies $f$, so TLPLan can backtrack
- Otherwise, need to determine whether there is an $M^+$ that satisfies $f^+$
  - For every action $a$ applicable to $s$,
    - » Let $s^+ = \gamma(s,a)$, and call TLPLan recursively on $f^+$ and $s^+$
- Next: how to compute $f^+$

**Procedure** Progress($f,s$)

- **Case:**
  1. $f$ contains no temporal ops : $f^+ :=$ TRUE if $s \models f$, FALSE otherwise
  2. $f = f_1 \wedge f_2$ : $f^+ :=$ Progress($f_1$, $s$) $\wedge$ Progress($f_2$, $s$)
  3. $f = f_1 \vee f_2$ : $f^+ :=$ Progress($f_1$, $s$) $\vee$ Progress($f_2$, $s$)
  4. $f = \neg f_1$ : $f^+ := \neg$Progress($f_1$, $s$)
  5. $f = \bigcirc f_1$ : $f^+ := f_1$
  6. $f = \Diamond f_1$ : $f^+ :=$ Progress($f_1$, $s$) $\vee f$
  7. $f = \Box f_1$ : $f^+ :=$ Progress($f_1$, $s$) $\wedge f$
  8. $f = f_1 \cup f_2$ : $f^+ :=$ Progress($f_2$, $s$) $\vee$ (Progress($f_1$, $s$) $\wedge f$)
  9. $f = \forall[x{:}g(x)]\ h(x)$ : $f^+ :=$ Progress($h_1$, $s$) $\wedge \ldots \wedge$ Progress($h_n$, $s$)
  10. $f = \exists[x{:}g(x)]\ h(x)$ : $f^+ :=$ Progress($h_1$, $s$) $\vee \ldots \vee$ Progress($h_n$, $s$)

  where $h_i$ is $h$ with $x$ replaced by the $i$'th element of $\{x : s \models g(x)\}$
- Next, simplify $f^+$ and return it
  - Boolean simplification rules:

    1. $[\text{FALSE} \wedge \phi | \phi \wedge \text{FALSE}] \mapsto \text{FALSE}$,
    2. $[\text{TRUE} \wedge \phi | \phi \wedge \text{TRUE}] \mapsto \phi$,
    3. $\neg\text{TRUE} \mapsto \text{FALSE}$,
    4. $\neg\text{FALSE} \mapsto \text{TRUE}$.

# Two Examples of ◯

- Suppose $f = \bigcirc on(\text{a,b})$
  - $f^+ = on(\text{a,b})$
  - $s^+$ is acceptable iff $on(\text{a,b})$ is true in $s^+$
- Suppose $f = \bigcirc\bigcirc on(\text{a,b})$
  - $f^+ = \bigcirc on(\text{a,b})$
  - $s^+$ is acceptable iff $\bigcirc on(\text{a,b})$ is true in $s^+$
    - » iff $on(\text{a,b})$ is true in $s^{++}$

**Case:**

| | | | |
|---|---|---|---|
| 1. | $f$ contains no temporal ops | : | $f^+ :=$ TRUE if $s \models f$, FALSE otherwise |
| 2. | $f = f_1 \wedge f_2$ | : | $f^+ :=$ Progress$(f_1, s) \wedge$ Progress$(f_2, s)$ |
| 3. | $f = f_1 \vee f_2$ | : | $f^+ :=$ Progress$(f_1, s) \vee$ Progress$(f_2, s)$ |
| 4. | $f = \neg f_1$ | : | $f^+ := \neg$Progress$(f_1, s)$ |
| 5. | $f = \bigcirc f_1$ | : | $f^+ := f_1$ |
| 6. | $f = \Diamond f_1$ | : | $f^+ :=$ Progress$(f_1, s) \vee f$ |
| 7. | $f = \Box f_1$ | : | $f^+ :=$ Progress$(f_1, s) \wedge f$ |
| 8. | $f = f_1 \cup f_2$ | : | $f^+ :=$ Progress$(f_2, s) \vee$ (Progress$(f_1, s) \wedge f$) |
| 9. | $f = \forall[x{:}g(x)]\ h(x)$ | : | $f^+ :=$ Progress$(h_1, s) \wedge \ldots \wedge$ Progress$(h_n, s)$ |
| 10. | $f = \exists[x{:}g(x)]\ h(x)$ | : | $f^+ :=$ Progress$(h_1, s) \vee \ldots \vee$ Progress$(h_n, s)$ |

# Example of ∧

- Suppose $f = on(\text{a,b}) \wedge \bigcirc on(\text{b,c})$
  - $f^+ = \text{Progress}(on(\text{a,b}), s) \wedge \text{Progress}(\bigcirc on(\text{b,c}), s)$
  - $\text{Progress}(on(\text{a,b}), s)$

    = TRUE if $on(\text{a,b})$ is true in $s$, else FALSE
  - $\text{Progress}(\bigcirc on(\text{b,c}), s) = on(\text{b,c})$
- If $on(\text{a,b})$ is true in $s$, then $f^+ = on(\text{b,c})$
  - i.e., $on(\text{b,c})$ must be true in $s^+$
- Otherwise, $f^+ = \text{FALSE}$
  - i.e., there is no acceptable $s^+$

**Case:**

| | | | |
|---|---|---|---|
| 1. | $f$ contains no temporal ops | : | $f^+ :=$ TRUE if $s \models f$, FALSE otherwise |
| 2. | $f = f_1 \wedge f_2$ | : | $f^+ := \text{Progress}(f_1, s) \wedge \text{Progress}(f_2, s)$ |
| 3. | $f = f_1 \vee f_2$ | : | $f^+ := \text{Progress}(f_1, s) \vee \text{Progress}(f_2, s)$ |
| 4. | $f = \neg f_1$ | : | $f^+ := \neg\text{Progress}(f_1, s)$ |
| 5. | $f = \bigcirc f_1$ | : | $f^+ := f_1$ |
| 6. | $f = \Diamond f_1$ | : | $f^+ := \text{Progress}(f_1, s) \vee f$ |
| 7. | $f = \Box f_1$ | : | $f^+ := \text{Progress}(f_1, s) \wedge f$ |
| 8. | $f = f_1 \cup f_2$ | : | $f^+ := \text{Progress}(f_2, s) \vee (\text{Progress}(f_1, s) \wedge f)$ |
| 9. | $f = \forall[x{:}g(x)]\ h(x)$ | : | $f^+ := \text{Progress}(h_1, s) \wedge \ldots \wedge \text{Progress}(h_n, s)$ |
| 10. | $f = \exists[x{:}g(x)]\ h(x)$ | : | $f^+ := \text{Progress}(h_1, s) \vee \ldots \vee \text{Progress}(h_n, s)$ |

# Example of $\Box$

- Suppose $f = \Box\, on(a,b)$
  - $f^+ = \text{Progress}(on(a,b), s) \wedge \Box\, on(a,b)$
- If $on(a,b)$ is true in $s$, then
  - $f^+ = \text{TRUE} \wedge \Box\, on(a,b) = \Box\, on(a,b) = f$
  - i.e., $on(a,b)$ must be true in $s^+, s^{++}, s^{+++}, \ldots$
- If $on(a,b)$ is false in $s$, then
  - $f^+ = \text{FALSE} \wedge \Box\, on(a,b) = \text{FALSE}$
  - There is no acceptable $s^+$

**Case:**

| | | | |
|---|---|---|---|
| 1. | $f$ contains no temporal ops | : | $f^+ :=$ TRUE if $s \models f$, FALSE otherwise |
| 2. | $f = f_1 \wedge f_2$ | : | $f^+ := \text{Progress}(f_1, s) \wedge \text{Progress}(f_2, s)$ |
| 3. | $f = f_1 \vee f_2$ | : | $f^+ := \text{Progress}(f_1, s) \vee \text{Progress}(f_2, s)$ |
| 4. | $f = \neg f_1$ | : | $f^+ := \neg\text{Progress}(f_1, s)$ |
| 5. | $f = \bigcirc f_1$ | : | $f^+ := f_1$ |
| 6. | $f = \Diamond f_1$ | : | $f^+ := \text{Progress}(f_1, s) \vee f$ |
| 7. | $f = \Box f_1$ | : | $f^+ := \text{Progress}(f_1, s) \wedge f$ |
| 8. | $f = f_1 \cup f_2$ | : | $f^+ := \text{Progress}(f_2, s) \vee (\text{Progress}(f_1, s) \wedge f)$ |
| 9. | $f = \forall[x{:}g(x)]\ h(x)$ | : | $f^+ := \text{Progress}(h_1, s) \wedge \ldots \wedge \text{Progress}(h_n, s)$ |
| 10. | $f = \exists[x{:}g(x)]\ h(x)$ | : | $f^+ := \text{Progress}(h_1, s) \vee \ldots \vee \text{Progress}(h_n, s)$ |

# Example of $\cup$

- Suppose $f = on(\text{a,b}) \cup on(\text{c,d})$
  - $f^+ = \text{Progress}(on(\text{c,d}), s) \vee (\text{Progress}(on(\text{a,b}), s) \wedge f)$
- If $on(\text{c,d})$ is true in $s$, then $\text{Progress}(on(\text{c,d}), s) = \text{TRUE}$
  - $f^+ = \text{TRUE}$, so any $s^+$ is acceptable
- If $on(\text{c,d})$ is false in $s$, then $\text{Progress}(on(\text{c,d}), s) = \text{FALSE}$
  - $f^+ = \text{Progress}(on(\text{a,b}), s) \wedge f$
  - If $on(\text{a,b})$ is false in $s$ then $f^+ = \text{FALSE}$: no $s^+$ is acceptable
  - If $on(\text{a,b})$ is true in $s$ then $f^+ = f$

**Case:**

1. $f$ contains no temporal ops : $f^+ := \text{TRUE if } s \models f, \text{ FALSE otherwise}$
2. $f = f_1 \wedge f_2$ : $f^+ := \text{Progress}(f_1, s) \wedge \text{Progress}(f_2, s)$
3. $f = f_1 \vee f_2$ : $f^+ := \text{Progress}(f_1, s) \vee \text{Progress}(f_2, s)$
4. $f = \neg f_1$ : $f^+ := \neg\text{Progress}(f_1, s)$
5. $f = \bigcirc f_1$ : $f^+ := f_1$
6. $f = \Diamond f_1$ : $f^+ := \text{Progress}(f_1, s) \vee f$
7. $f = \Box f_1$ : $f^+ := \text{Progress}(f_1, s) \wedge f$
8. $f = f_1 \cup f_2$ : $f^+ := \text{Progress}(f_2, s) \vee (\text{Progress}(f_1, s) \wedge f)$
9. $f = \forall[x{:}g(x)]\ h(x)$ : $f^+ := \text{Progress}(h_1, s) \wedge \ldots \wedge \text{Progress}(h_n, s)$
10. $f = \exists[x{:}g(x)]\ h(x)$ : $f^+ := \text{Progress}(h_1, s) \vee \ldots \vee \text{Progress}(h_n, s)$

# Another Example

- Suppose $f = \Box(on(\text{a,b}) \Rightarrow \bigcirc clear(\text{a}))$
  - $f^+ = \text{Progress}[on(\text{a,b}) \Rightarrow \bigcirc clear(\text{a}), s] \wedge f$
    $= (\neg\text{Progress}[on(\text{a,b})] \vee clear(\text{a})) \wedge f$
  - If $on(\text{a,b})$ is false in $s$, then $f^+ = (\text{TRUE} \vee clear(\text{a})) \wedge f = f$
    - » So $s^+$ must satisfy $f$
  - If $on(\text{a,b})$ is true in $s$, then $f^+ = clear(\text{a}) \wedge f$
    - » So $s^+$ must satisfy both $clear(\text{a})$ and $f$

**Case:**

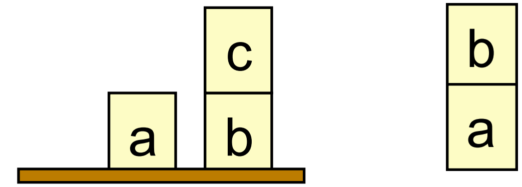
1. $f$ contains no temporal ops : $f^+ :=$ TRUE if $s \models f$, FALSE otherwise
2. $f = f_1 \wedge f_2$ : $f^+ := \text{Progress}(f_1, s) \wedge \text{Progress}(f_2, s)$
3. $f = f_1 \vee f_2$ : $f^+ := \text{Progress}(f_1, s) \vee \text{Progress}(f_2, s)$
4. $f = \neg f_1$ : $f^+ := \neg\text{Progress}(f_1, s)$
5. $f = \bigcirc f_1$ : $f^+ := f_1$
6. $f = \Diamond f_1$ : $f^+ := \text{Progress}(f_1, s) \vee f$
7. $f = \Box f_1$ : $f^+ := \text{Progress}(f_1, s) \wedge f$
8. $f = f_1 \cup f_2$ : $f^+ := \text{Progress}(f_2, s) \vee (\text{Progress}(f_1, s) \wedge f)$
9. $f = \forall[x{:}g(x)]\ h(x)$ : $f^+ := \text{Progress}(h_1, s) \wedge \ldots \wedge \text{Progress}(h_n, s)$
10. $f = \exists[x{:}g(x)]\ h(x)$ : $f^+ := \text{Progress}(h_1, s) \vee \ldots \vee \text{Progress}(h_n, s)$

# Pseudocode for TLPlan

- Nondeterministic forward search
  - Input includes a control formula $f$ for the current state $s$
  - If $f^+$ = FALSE then $s$ has no acceptable successors => backtrack
  - Otherwise the progressed formula is the control formula for $s$'s children

```
Procedure TLPlan (s, f, g, π)
        if f = FALSE then return failure
        if s satisfies g then return π
        f⁺ ← Progress (f, s)
        if f⁺ = FALSE then return failure
        A ← {actions applicable to s}
        if A is empty then return failure
        nondeterministically choose a ∈ A
        s⁺ ← γ(s,a)
        return TLPlan (s⁺, f⁺, g, π.a)
```

# Example Planning Problem

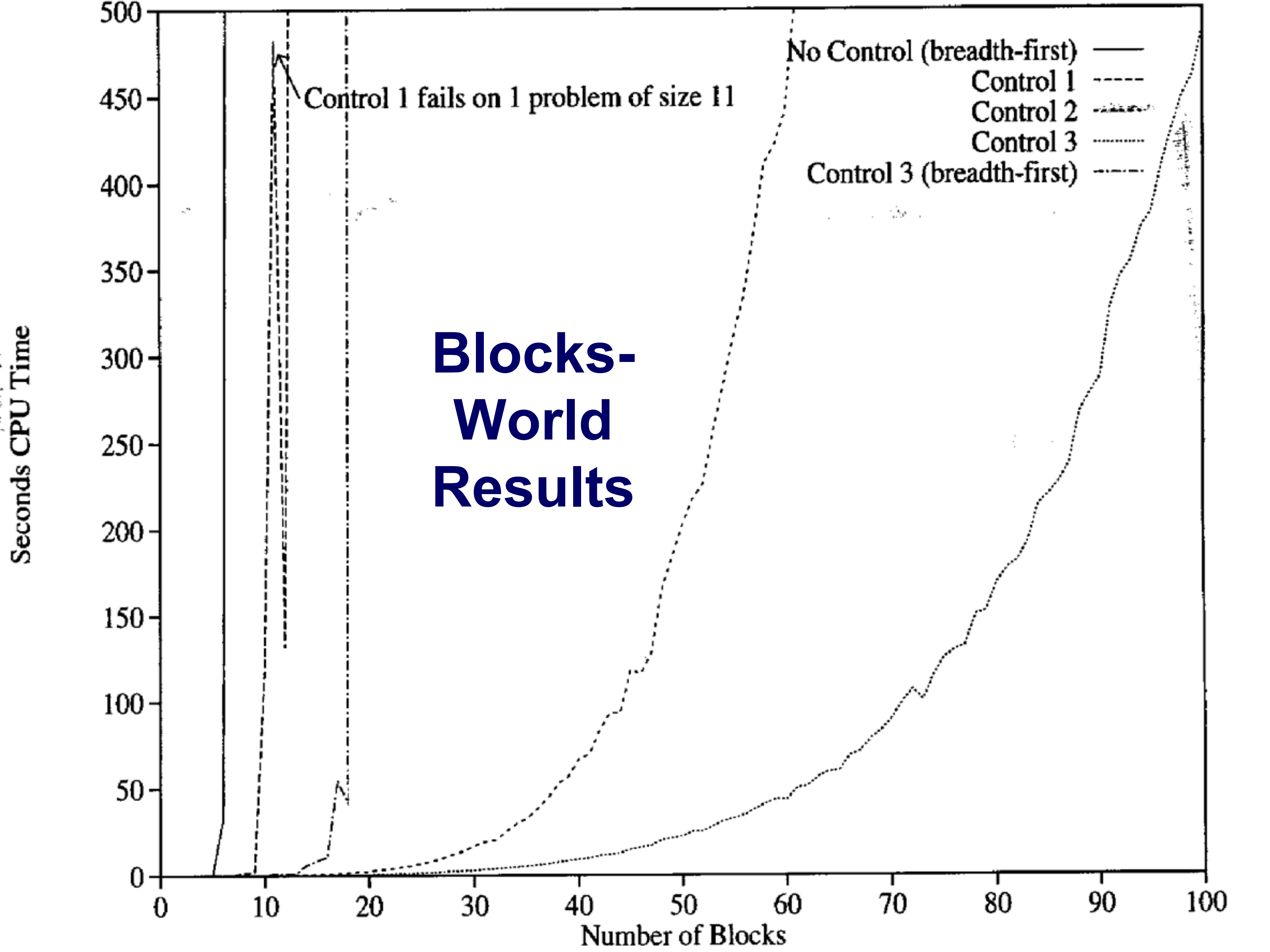



- $s = \{ontable(\mathsf{a}), ontable(\mathsf{b}), clear(\mathsf{a}), clear(\mathsf{c}), on(\mathsf{c},\mathsf{b})\}$
- $g = \{on(\mathsf{b}, \mathsf{a})\}$
- $f = \Box\forall[x{:}clear(x)]\ \{(ontable(x) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(x,y))]) \Rightarrow \bigcirc\neg holding(x)\}$
  - never pick up a block $x$ if $x$ is not required to be on another block $y$

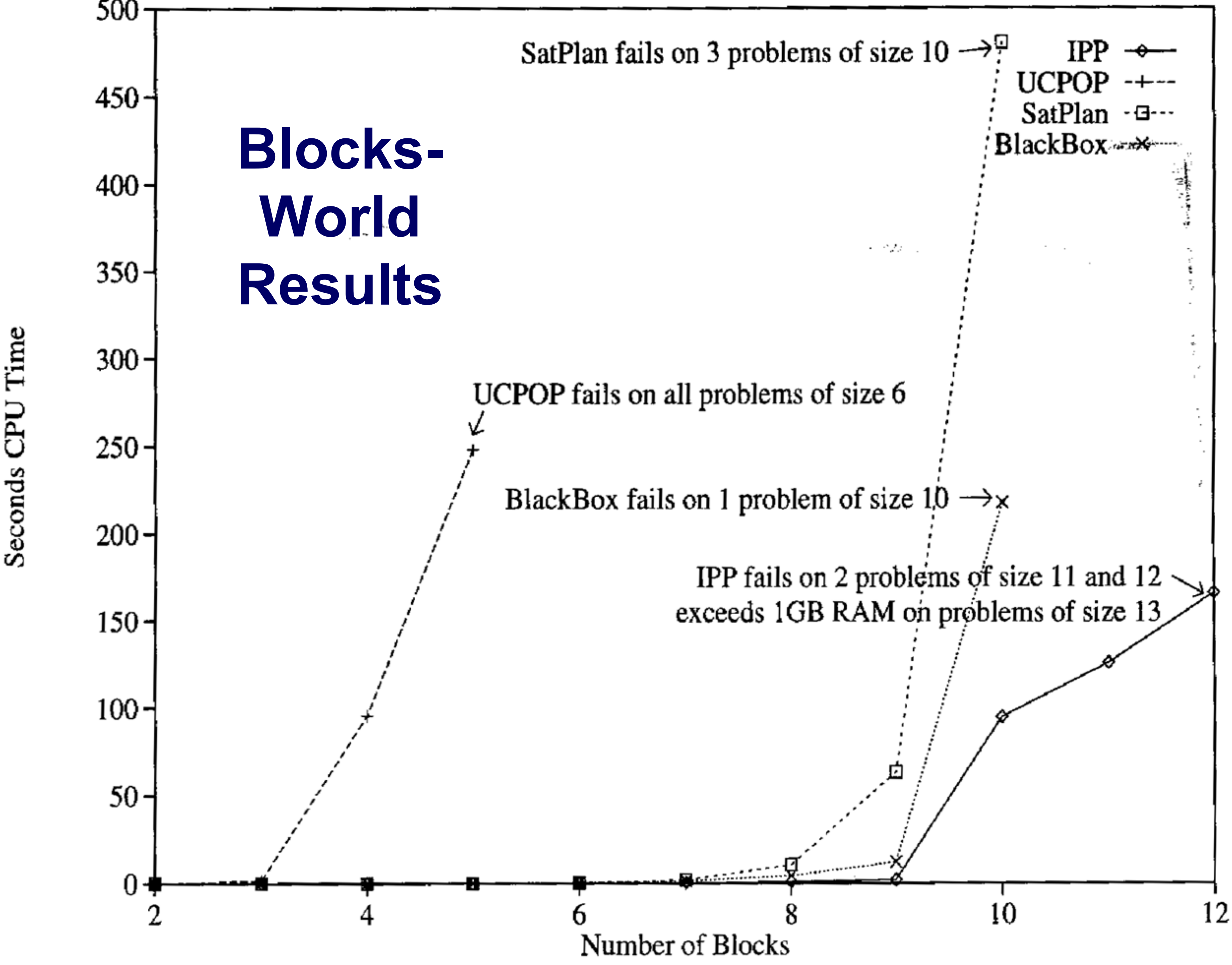
- $f^+ = \mathrm{Progress}(f_1,s) \wedge f$, where
  - $f_1 = \forall[x{:}clear(x)]\{(ontable(x) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(x,y))]) \Rightarrow \bigcirc\neg holding(x)\}$
- $\{x{:}\ clear(x)\} = \{\mathsf{a}, \mathsf{c}\}$, so

$$\begin{aligned}\mathrm{Progress}(f_1,s) = {} & \mathrm{Progress}((ontable(\mathsf{a}) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(\mathsf{a},y))]) \Rightarrow \bigcirc\neg holding(\mathsf{a})\},s) \\ & \wedge \mathrm{Progress}((ontable(\mathsf{c}) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(\mathsf{c},y))]) \Rightarrow \bigcirc\neg holding(\mathsf{b})\},s) \\ = {} & (\mathrm{TRUE} \Rightarrow \neg holding(\mathsf{a})) \wedge \mathrm{TRUE} = \neg holding(\mathsf{a})\end{aligned}$$

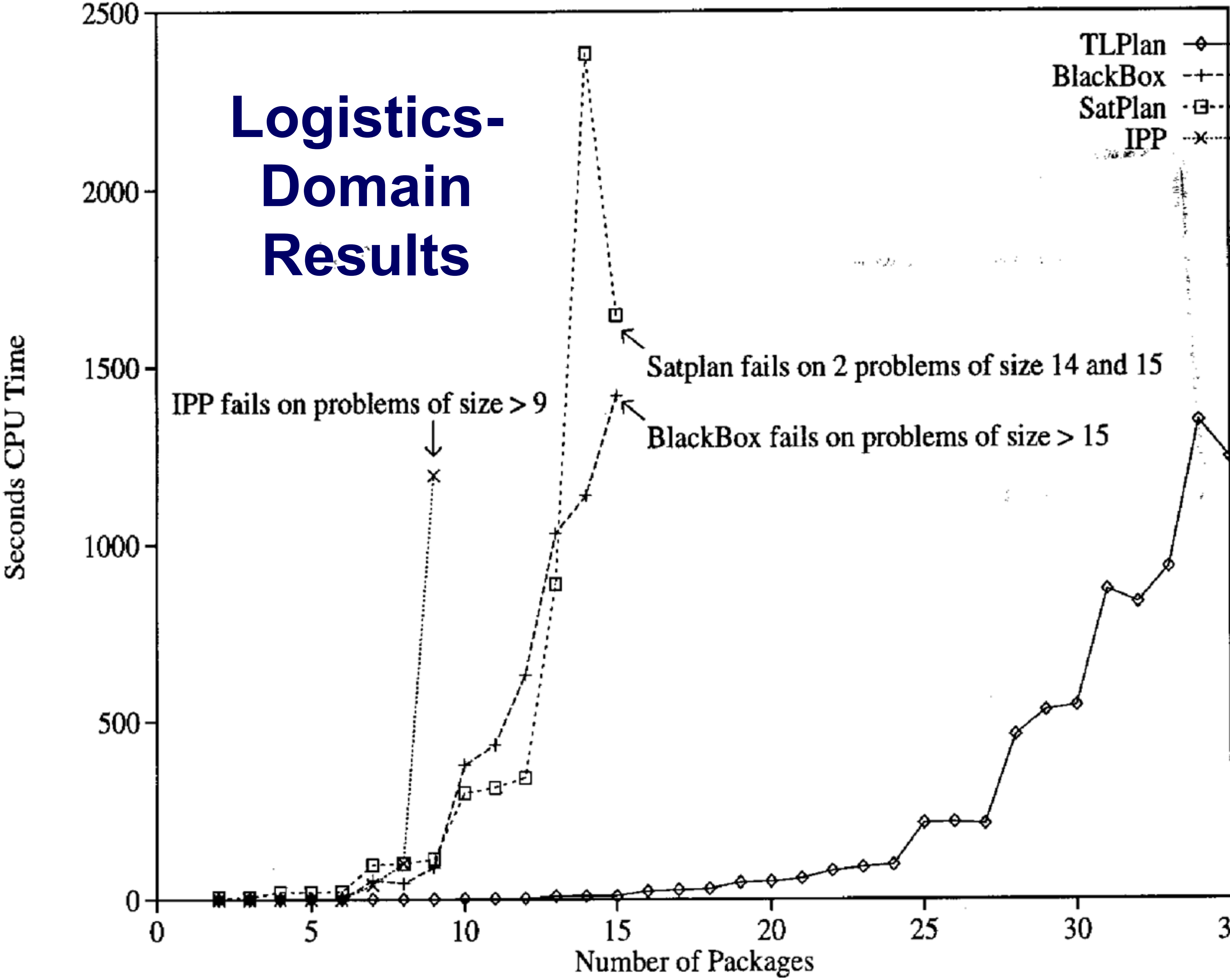
- $f^+ = \neg holding(\mathsf{a}) \wedge f$
  $= \neg holding(\mathsf{a}) \wedge$
  $\Box\forall[x{:}clear(x)]\ \{(ontable(x) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(x,y))]) \Rightarrow \bigcirc\neg holding(x)\}$
- Two applicable actions: pickup(a) and pickup(c)
  - Try $s^+ = \gamma(s, \text{pickup}(\mathsf{a}))$: $f^+$ simplifies to FALSE $\Rightarrow$ backtrack
  - Try $s^+ = \gamma(s, \text{pickup}(\mathsf{c}))$: $f^+$ doesn't simplify to FALSE $\Rightarrow$ keep going


Blocks-World Results
No Control (breadth-first)
Control 1
Control 2
Control 3
Control 3 (breadth-first)
Control 1 fails on 1 problem of size 11
Seconds CPU Time
Number of Blocks
0
50
100
150
200
250
300
350
400
450
500
0
10
20
30
40
50
60
70
80
90
100

# Blocks-World Results



# Logistics-Domain Results



# Discussion

- 2000 International Planning Competition
  - TALplanner: similar algorithm, different temporal logic
    - » received the top award for a "hand-tailored" (i.e., domain-configurable) planner
- TLPlan won the same award in the 2002 International Planning Competition
- Both of them:
  - Ran several orders of magnitude faster than the "fully automated" (i.e., domain-independent) planners
    - » especially on large problems
  - Solved problems on which the domain-independent planners ran out of time or memory