Automated Planning

Complexity

Jonas Kvarnström
Automated Planning Group
Department of Computer and Information Science
Linköping University

Complexity of Classical Planning

Complexity 1

- What is the complexity of plan generation?
  - How much time and space (memory) do we need?

- **Vague question** – let's try to be more specific…
  - Assume an **infinite set** of problem instances
    - For example, "all classical planning problems"
  - Analyze all possible algorithms in terms of **asymptotic worst case complexity**
  - What is the **lowest** worst case complexity we can achieve?

Complexity 2: Repetition

- What is asymptotic complexity?
  - An algorithm is in <u>O(f(n))</u> if there is an algorithm for which there exists a **<u>fixed constant c</u>** such that for **<u>all n</u>**, the time to **<u>solve an instance of size n</u>** is **<u>at most c * f(n)</u>**
- Example: Sorting is in O(n log n),
  - We can find an algorithm for which there is a **<u>fixed constant c</u>** such that for **<u>all n</u>**, the time to **<u>sort n elements</u>** is **<u>at most c * n log n</u>**

So sorting is **<u>also</u>** in $O(n^2)$, in $O(2^n)$, and so on.

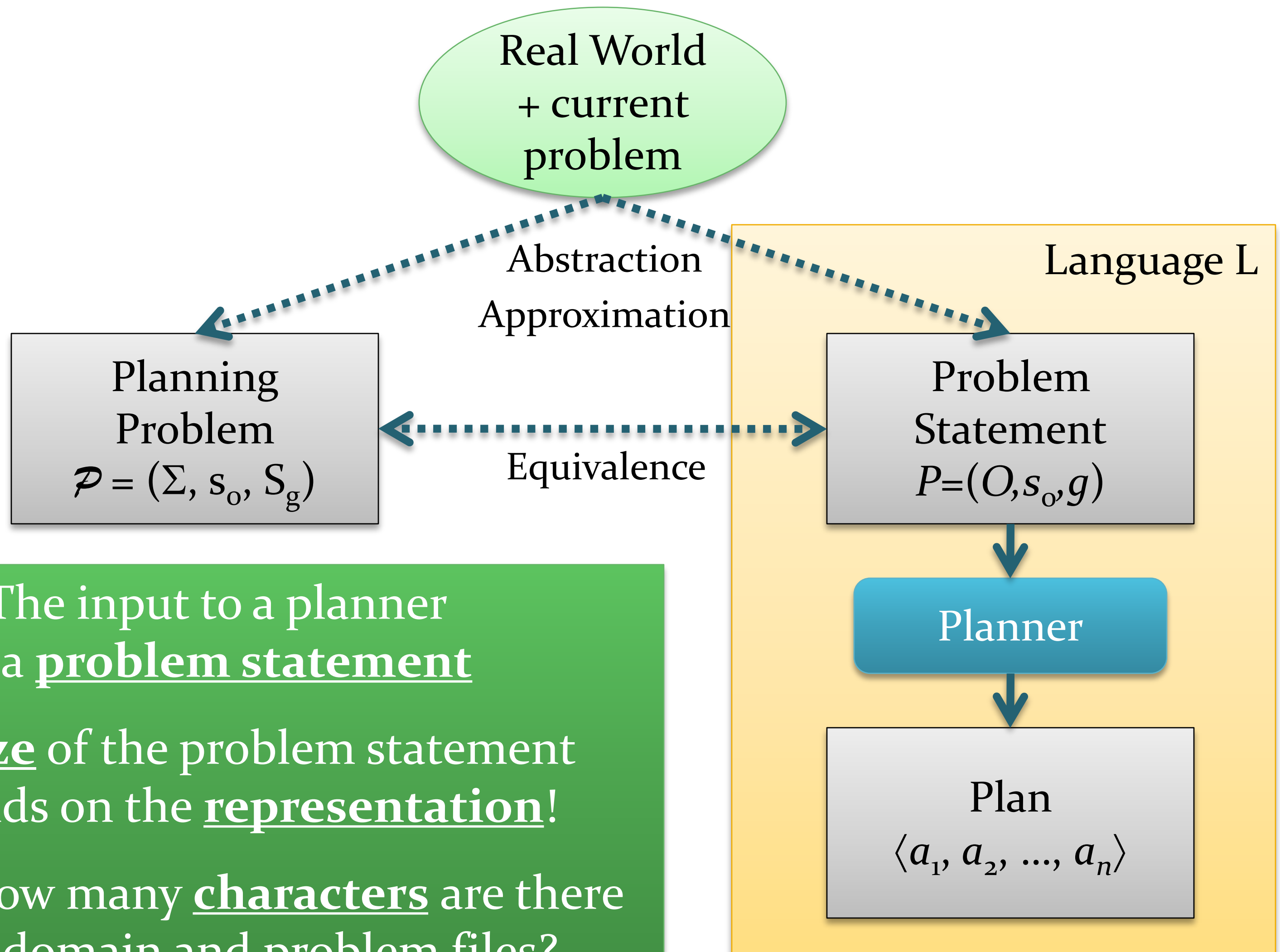
If we can do it in "at most n log n", we can do it in "at most n^2".

<u>Some</u> problem instances might be solved faster!
If the list happens to be sorted already, we might finish in *linear* time: O(n)

But for the entire **<u>set</u>** of problems, our guarantee is O(n log n)

Size of a Planning Problem

So what is "a planning problem of **size** n"?



The input to a planner
is a **problem statement**

The **size** of the problem statement
depends on the **representation**!

PDDL: How many **characters** are there
in the domain and problem files?

Complexity of Planning

- Now: The complexity of PLAN-EXISTENCE
  - The problem of finding out whether there **exists** a **solution**

- We will be satisfied with a "rough" classification
  - Polynomial, exponential, …
  - Some common complexity **classes**:
    - NLOGSPACE
      $\subseteq$ P = the algorithms that can be executed in polynomial time
      $\subseteq$ NP
      $\subseteq$ PSPACE
      $\subseteq$ EXPTIME = can be executed in exponential time
      $\subseteq$ NEXPTIME
      $\subseteq$ EXPSPACE = the algorithms that can be executed in exponential space
  - Example: P = PTIME = polynomial time
    - May require n^2 time, or n^{10}, or $n^{1000000}$
    - For large enough problems, $n^{1000000} < 1.001^n$
    - Sorting is in P, and therefore also in NP, PSPACE, …

Complexity Analysis: Observations

- Most representations use:
  - **Operators** that have **parameters** and many **instances** (called actions)
  - **Predicates** that have parameters and many instances
  - Consider an untyped problem of size 1000, with 100 constants (objects)
    - One action: DoIt(?a,?b) – 10,000 instances
    - One predicate: pred(?a,?b) – 10,000 instances
  - Now add more parameters to the operator / predicate:
    - DoIt(?a,?b,?c) – 1,000,000 instances, problem size 1001
    - DoIt(?a,?b,?c,?d) – 100,000,000 instances, problem size 1002
  - **Adding** 3 characters **multiplies** the number of instances by 100

In the worst case, the number of actions / predicate instances is **exponential** in the size of the domain definition!

Complexity Analysis: Classical 1

First Problem Set:
All **planning problem statements** in the **classical representation**

- How can we analyze this case?
  - |A| is at most exponential (number of actions)
  - But a plan might have to use the same action many times
  - Difficult to find a bound on plan length…
  - Let's try another approach

NLOGSPACE
$\subseteq$ P
$\subseteq$ NP
$\subseteq$ PSPACE
$\subseteq$ EXPTIME
$\subseteq$ NEXPTIME
$\subseteq$ EXPSPACE

Complexity Analysis: Classical 2

First Problem Set:
All **planning problem statements** in the **classical representation**

- How can we analyze this case?
  - Visiting all reachable states would be sufficient
  - We have **at most** an exponential number of states
    - Even if our enemies try as much as possible
      to use every increase in problem size
      to make the problem harder
  - ➔ Keeping track of which states we have visited cannot take more than exponential space
  - ➔ Plan existence cannot be harder than EXPSPACE
    - In fact, EXPSPACE-*complete* (Won't prove it here…)

NLOGSPACE
$\subseteq$ P
$\subseteq$ NP
$\subseteq$ PSPACE
$\subseteq$ EXPTIME
$\subseteq$ NEXPTIME
$\subseteq$ EXPSPACE

Complexity Analysis: Classical 3

Second Problem Set:
All planning problem statements in the **classical representation** that only have **positive effects** (but pos+neg preconditions allowed)

- Only positive effects
  - ➔ The set of **true facts** increases monotonically as new actions are added
  - ➔ There can be no point in applying the same action twice!
  - (But action order matters, due to negative preconditions)
- Checking **every** sequence of **unique** actions would be sufficient
  - We have at most an exponential number of actions
  - ➔ A plan can be at most exponentially long
- Non-deterministic algorithms can (conceptually) "test all alternatives at once",
  - ➔ in NEXPTIME
  - (Actually, NEXPTIME-complete)

NLOGSPACE
$\subseteq$ P
$\subseteq$ NP
$\subseteq$ PSPACE
$\subseteq$ EXPTIME
$\subseteq$ NEXPTIME
$\subseteq$ EXPSPACE

Complexity Analysis: Classical 4

Third Problem Set:

All planning problem statements in the **classical representation** that only have **positive effects** and **positive preconditions**

- Only positive effects
  - ➔ The set of **true facts** increases monotonically as new actions are added
- Only positive effects **and** only positive preconditions
  - ➔ The set of **applicable actions** increases monotonically

- ➔ Action order does not matter!
  - If you can apply A1 now, you can apply A1 after any other actions as well
  - **Could** just apply **all** actions until we reach a fixpoint
  - If the goal is satisfied in the final state, there exists a plan
  - Exponential number of actions ➔ in EXPTIME
  - (Actually, it is EXPTIME-*complete*!)

NLOGSPACE
$\subseteq$ P
$\subseteq$ NP
$\subseteq$ PSPACE
$\subseteq$ EXPTIME
$\subseteq$ NEXPTIME
$\subseteq$ EXPSPACE

Complexity Analysis: Classical 5

- One reason for high complexity:
  Operators can be modified as n increases
  - Suppose **operators** are **fixed / given in advance**!
    - They are not part of the problem statement, cannot be changed
    - We can only increase n by changing the problem instance:
      objects, **initial state** and **goal**
- For the **classical** representation:
  - Arbitrary classical problem:
    - EXPSPACE-complete ➔ PSPACE
  - Only **positive effects**:
    - NEXPTIME-complete ➔ NP or NP-complete, depending on the operators
  - Only **positive effects**, only **positive preconditions**:
    - EXPTIME-complete ➔ P

These results are generally more relevant!
We are usually interested in what happens with more objects,
not if we change operators in the "worst" way possible

NLOGSPACE
$\subseteq$ P
$\subseteq$ NP
$\subseteq$ PSPACE
$\subseteq$ EXPTIME
$\subseteq$ NEXPTIME
$\subseteq$ EXPSPACE

Complexity Analysis: Domains

- Note: This complexity applies to the **worst case**
  - We saw that restricting the set of problems gives us tighter time bounds

Handle **all** planning problem statements in the **classical representation** (with pos+neg effects and pos+neg preconditions)
→ EXPSPACE-complete

Handle **all** planning problem statements in the **classical representation** that only have **positive effects** and **positive preconditions**
→ EXPTIME-complete

Handle **all** planning problem statements for the **standard blocks world**
→ P (polynomial time *given an optimal algorithm*)