6th International Planning Competition: Uncertainty Part

Daniel Bryce
SRI International
bryce@ai.sri.com

Olivier Buffet
LORIA-INRIA
olivier.buffet@loria.fr

Abstract

The 6th International Planning Competition will be co-located with ICAPS-08 in Sydney, Australia. This competition will contain three parts: i) the classical part, ii) the uncertainty part and iii) the learning part. This document presents the uncertainty part and its various tracks. The official site for the uncertainty part is: http://ippc-2008.loria.fr/wiki/, where this document and other materials are provided.

Introduction

The 6th International Planning Competition (IPC-6) will be co-located with the 18th International Conference on Automated Planning and Scheduling (ICAPS-08) in Sydney, Australia on September 14–18, 2008. The competition is a biannual event where a number of planning systems are evaluated on a variety of problems. This is an opportunity to compare existing algorithms and to provide the automated planning community with recognized benchmarks written in a standardized language: PDDL.

The IPC started in 1998, taking place at AIPS until 2004, then at ICAPS. It was originally limited to classical planning, i.e. automated planning in deterministic domains. In 2004 has been introduced a "probabilistic track" —organized by Michael Littman and Haakan Younes— which introduced an extension of PDDL to probabilistic domains and a client-server plan evaluator (MDPSim). In 2006 this track —organized by Blai Bonet and Bob Givan— has been enriched with a conformant subtrack and renamed "non-deterministic track".

This year's competition (2008) renames "tracks" as "parts" and includes three of them:

- the classical part —organized by Minh Do, Malte Helmert and Ioannis Refanidis[1]—,
- the uncertainty part[2] —organized by ourselves—, and
- a novel learning part —organized by Alan Fern, Roni Khardon and Prasad Tadepalli[3]—, in which planners can train on small instances of problems before being evaluated on large ones.

This document is a modified version of the previous edition's call for participations (Bonet & Givan 2005). It presents the uncertainty part, which is made of

- a non-observable non-deterministic (conformant) track,
- a fully-observable non-deterministic track,
- a fully-observable probabilistic track and
- a partially-observable probabilistic track.

Planning Tasks and Solutions

Most of the competition will focus on shortest-path planning problems. This is more general than goal reachability with unit costs as actions may have a non unit cost. Problems of this type can be described by models of the form:

M1. a finite state space (set of states) S,

M2. an initial state $s_0 \in S$,

M3. a set $S_G \subseteq S$ of goal states,

M4. sets $A(s)$ of applicable actions for each $s \in S$,

M5. a cost function $c(s, a) \rightarrow \mathcal{R}$, and

M6. a non-deterministic transition function $F(s, a) \subseteq S$.

Models of M1–M6 are described using a high-level planning language based on propositional logic in which the states are valuations for the propositional symbols, the set of initial and goal states are described by logical formulae, and the set of applicable actions (operators) and the transition function are described by means of action schemata.

The form of a solution and the optimality criteria depend on the particular planning task as follows.

Non-Observable Non-Deterministic Planning (NOND/Conformant Planning)

The problem of conformant planning is that of deciding whether there exists a linear sequence of actions that will achieve the goal from any initial state and any resolution of the non-determinism in the problem (Goldman & Boddy 1996; Smith & Weld 1998).

In non-observable or partially observable domains, *belief states* are used to represent ones belief of the possible current states. The belief state at time-step n depends on the

[1] http://ipc.informatik.uni-freiburg.de/

[2] "Uncertainty" is preferred over "probabilistic" or "non-deterministic" to avoid possible confusions.

[3] http://eecs.oregonstate.edu/ipc-learn/

initial belief state b_0 and on the history of past observations (if any) and actions ($h_n = \langle a_0, o_0, \ldots, a_{n-1}, o_{n-1}\rangle$) since the initial state.[4]

In a non-deterministic setting, a *non-deterministic*[5] *belief state* b is a set of states. Then, a conformant planning problem is modeled with M1–M6, where M2 and M5 are redefined as

M2'. an initial belief state $b_0 \subseteq S$, and

M5'. a cost function $c(s, a) \rightarrow 1$.

Given this model, we say that $s_0, a_0, \ldots, a_{n-1}, s_n$ is a trajectory generated by actions $a_0, \ldots, a_{n-1}$ when

C1. $s_0 \in b_0$,

C2. $a_k \in A(s_k)$ for $0 \leq k < n$, and

C3. $s_{k+1} \in F(s_k, a_k)$ for $0 \leq k < n$.

The plan $a_0, \ldots, a_{n-1}$ is a (valid) solution to the model if each trajectory under $a_0, \ldots, a_{n-1}$ is such that $s_n \in S_G$. A valid plan π is assigned a (worst-case scenario) cost V_π equal to the longest trajectory starting in some $s_0 \in b_0$ and ending at a goal state.

The plan is optimal if its value is minimal.

Fully-Observable Non-Deterministic Planning (FOND Planning)

Non-deterministic planning with full observability refers to deciding whether there exists a conditional plan that achieves the goal for a model satisfying M1–M6, where M5 is redefined as

M5'. a cost function $c(s, a) \rightarrow 1$.

The main difference from conformant planning is that solutions are policies (partial functions) mapping states into actions, rather than linear sequences of operators.

Let $\pi : S \rightarrow \bigcup_{s \in S} A(s)$ be a policy for model M1–M6, S_π the domain of definition of π, and $S_\pi(s)$ the set of states *reachable* from s using π, then we say that:

a) π is *closed* with respect to s iff $S_\pi(s) \subseteq S_\pi$,

b) π is *proper* with respect to s iff a goal state can be reached using π from all $s \in S_\pi(s)$,

c) π is *acyclic* with respect to s iff there is no trajectory $s = s_0, \pi(s_0), \ldots, s_i, \ldots, s_j, \ldots, s_n$ with i and j such that $0 \leq i < j \leq n$, and $s_i = s_j$.

d) π is *closed* (resp. *proper* or *acyclic*) with respect to $S' \subseteq S$ if it is closed (resp. proper or acyclic) with respect to all $s \in S'$,

A policy π is a valid solution for the non-deterministic model iff π is closed and proper with respect to the initial state s_0 . A valid policy π is assigned a (worst-case scenario) cost V_π equal to the longest trajectory starting at s_0 and ending at a goal state. For acyclic policies with respect to s_0, the cost V_π is always well defined, i.e. $< +\infty$.

[4]We make the assumption that there is no observation prior to the first action.

[5]"non-deterministic" is usually omitted when the context is clear.

A policy π is optimal for model M1–M6 if it is a valid solution of minimum V_π value.

The competition will only judge the cost of plans in non-deterministic domains that admit acyclic solutions, where optimal solutions always have finite cost. In non-deterministic domains with cyclic solutions, the solutions will be judged solely by the time taken to generate a solution.

Fully-Observable Probabilistic Planning (FOP Planning)

Probabilistic planning problems —here stochastic shortest-path problems— can be described by models M1–M6 extended with

M7. transition probabilities $0 < P_a(s'|s)$, for $s' \in F(s, a)$ and $a \in A(s)$, such that $\sum_{s' \in F(s,a)} P_a(s'|s) = 1$.

In this case, solutions are also policies π that map states into actions. As in the fully-observable non-deterministic case, definitions (a)–(d) can be used to characterize the properties of π. A policy π is a valid solution if it is closed and proper with respect to s_0. The cost V_π assigned to a valid π is defined as the expected cost incurred by the policy when it is applied from the initial states, i.e. V_π is defined as $V_\pi(s_0)$ where the function $V_\pi(\cdot)$ is the unique solution to the Bellman equation giving $V_\pi(s)$ for states $s \notin S_G$:

$$V_\pi(s) = \sum_{a \in A(s)} c(s, a) + \sum_{s' \in F(s,\pi(s))} P_a(s'|s) V_\pi(s'),$$

where $V_\pi(s)$ is taken to be zero for $s \in S_G$. We can then take as optimal any policy for a probabilistic model M1–M6 that is a valid policy with minimum V_π value.

Note that we use a formulation with costs rather than rewards in this section to remain in a cost-minimization problem in all tracks. This doesn't prevent some costs to be negative. Planning domains will be adapted so that a large reward (resp. cost) will be associated to each success (resp. each failure).

Partially-Observable Probabilistic Planning (POP Planning)

Partially observable probabilistic planning problems can be described by models M1–M7 with the following changes:

M2'. an initial belief state $b_0 \in \Pi(S)$,

M3'. a set $S_G \subseteq S$ of goal states *recognizable through their observations*,

M4'. sets $A(s)$ of applicable actions for each $s \in S$

and extended with

M8. a set Ω of possible observations and

M9. an observation function $O(o|s, a)$ giving the probability to get observation o in state s and given the last action.

In this probabilistic setting, let us define a new type of belief state. A *(probabilistic) belief state* b is a probability distribution over states ($b(s) = Pr(s)$). The current belief state at time-step n can be written as: $b_n(\cdot) = Pr(\cdot|b_0(\cdot), h_n)$. For $n > 0$ and given the last action-observation pair $\langle a, o\rangle =$

$\langle a_{n-1}, o_{n-1}\rangle$, the belief state is updated with (for all $s' \in S$):

$$b_n(s') = \frac{O(o|s',a)\sum_{s\in S} Pr(s'|s,a)b_{n-1}(s)}{\sum_{s''} O(o|s'',a)\sum_{s\in S} Pr(s''|s,a)b_{n-1}(s)}.$$

In this case, solutions are policies that map belief states into actions. With some notational changes, definitions (a)–(d) can be used to characterize the properties of π. A policy π is a valid solution if it is closed and proper with respect to S_0. The cost V_π assigned to a valid π is defined as the expected cost incurred by the policy when it is applied from the initial states, i.e. V_π is defined as $V_\pi(b_0)$ where the function $V_\pi(\cdot)$ is the unique solution to the Bellman equation giving $V_\pi(b)$ for belief states $b \notin B_G$[6]:

$$V_\pi(b) = \sum_{s\in S, a\in A(o)} b(s)\left[c(s,a) + \sum_{b'\in B} P_a(b'|s)V_\pi(b')\right],$$

where $V_\pi(b)$ is taken to be zero for $b \in B_G$. We can then take as optimal any policy for a probabilistic model M1–M3'-M4'–M10 that is a valid policy with minimum V_π value.

Languages

Modeling Languages

For the first three tracks of the IPC (all but the POP one), there is no need to introduce a new description language or to significantly modify PPDDL. Thus, the official language for the competition is PPDDL with minor extensions required to model non-deterministic effects. On the other hand, the original PPDDL specification is too ample for the competition needs, and thus only a subset of it will be actually used. The formal definition of PPDDL and its semantics is given in (Younes & Littman 2004).

Future editions of the IPC should probably aim at using an extension of PDDL for the POP track as well. This is not the case this year for two reasons: 1) there is no common agreement on how to extend PDDL and 2) properly extending MDPSim would probably take too much time. The result is that this edition's POP track the modeling language that will be used is Cassandra's file format.

This section describes the extensions and subset of 1) PPDDL and 2) Cassandra's file format to be used in the competition as well as the output language.

All Tracks (but POP) For the competition, PPDDL specifications will be based on the `:adl` requirement, i.e. STRIPS with arbitrary conditions and conditional effects, yet no existential quantification, disjunctions or negative literals will be permitted in the preconditions of operators nor in the conditions for conditional effects. However, general formulae will be allowed in the descriptions of the goals. As mentioned in the PPDDL manual, all effects will be order independent and non-conflicting (interfering), see (Younes & Littman 2004) for details.

[6] B_G is the set of "goal belief states". But, as goal states are supposed to be immediately distinguishible through instant observations, perceiving such a "goal observation" ensures that $b(s) = 0$ for all $s \notin S_G$.

Non-Deterministic Tracks We extend the formal definition of PPDDL with an additional non-deterministic statement, the counterpart of the `probabilistic` statement for non-deterministic models, of the form:

$$(\texttt{oneof}\ e_1\ e_2\ \ldots\ e_n)$$

where the e_k's are PPDDL effects. We also use the requirement tag `non-deterministic` for such domains. The semantics is that, when executing such effect, one of the e_i is chosen and applied to the current state. The same statements will be used to express the set of initial states, treating the initial state as the effect of a "dummy" start action.

Partially-Observable Probabilistic Track Contrary to PDDL, Cassandra's file format does not make possible to encode a lot of structure of the planning problem. All states, actions and observations are enumerated (specified with numbers or names). For example, here are two possible definitions of 4 "move" actions:

```
actions: 4
actions: north south east west
```

The reward (/cost) and transition function benefit from special constructs that allow for a compact encoding of many domains. You can for example specify that action 5 always sends you to state 0 by writing:

```
T: 5 : * : 0 1.0
```

A full description of the file format along with a grammar and examples can be found on Anthony (Tony) Cassandra's POMDP page: `http://pomdp.org/pomdp/`

The only constraint we are putting in the competition is that all problems will be stochastic-shortest path problems.

Appendix A makes a summarized description of the versions of the PPDDL language to be used in the competition.

Output Languages and Evaluation

For those tasks where an explicit solution plan is required—here: non-deterministic planning—a language for describing such plans is required. For other tasks—here: probabilistic planning—plans (policies) are evaluated by Monte-Carlo simulation using the MDPSim simulator.

Non-Deterministic Tracks In both non-deterministic tracks, only proper plans are considered as valid. Thus, non-deterministic planners will be required to output the solution policy into a file in a suitable representation language so that this file can be passed on to a plan verifier. This section describes such language. Other output languages will be considered on request, as needed, but the competition staff may not have the resources to support additional output languages. If the properness verifier detects an improper policy, then the planner is disqualified from the corresponding instance, and may at the organizers' discretion be disqualified from the entire track.

The competition software will not only check plans for properness, but also compute their cost. At the end, the planners are ranked by quality of solution and time to compute. For this case, we plan to distribute the properness verifier and cost computation software so competitors can prepare

in advance. The software will read the PPDDL specification of the problem and the plan description, and then output whether the plan is proper and its cost.

In the output language, the file contains three sections separated by '%%':

```
<n> <atom-list>
%%
<m> <action-list>
%%
<plan>
```

where <n> is an integer, possibly 0, denoting the size of <atom-list> which is a space-separated list of atoms such as '(on A B)', <m>, possibly 0, is the size of <action-list> which is a space-separated list of operators 'such as (move A C B)', and <plan> is the representation of the plan.

For **conformant planning** problems, <plan> must be of the form:

```
linear <k> <integer-list>
```

where <k> is the size of <integer-list> which is a space-separated list of integers in $[0, m-1]$ each denoting the action with such index.

For example, the following file denotes the conformant plan '(pick B);(putdown B)':

```
0
%%
4 (pick A) (putdown A) (pick B) (putdown B)
%%
linear 2 2 3
```

For **fully-observable non-deterministic** problems, <plan> can be either an explicit or a factored representation of the policy. In the first case, <plan> is of the form:

```
policy <k> <map-list>
```

where <k> is the size of <map-list> which is a space-separated list of variable-sized elements. The elements of <map-list> define a partial function mapping states into actions. Each element is of the form:

```
<l> <atom-list> <action-index>
```

where <l> is the size of <atom-list> which is a space-separated list of integers in $[0, n-1]$, each denoting the atom with such index, and <action-index> is an integer in $[0, m-1]$ denoting the action with such index. Such element defines the mapping from the unique state that makes all and only all atoms in <atom-list> true into the action with appropriate index.

For example, the file:

```
4 (on A B) (clear A) (clear B) (on B A)
%%
4 (pick A) (putdown A) (pick B) (putdown B)
%%
policy 2 2 0 1 0 2 3 2 2
```

denotes the policy π such that $\pi(s)$ and $\pi(s')$ are '(pick A)' and '(pick B)' respectively, and s = {(on A B), (clear A)} and s' = {(on B A), (clear B)}.

Factored representation of policies are supported in the form of Free Algebraic Decision Diagrams (FADDs) (Bryant 1992). An FADD is like an Free Binary Decision Diagram whose leaves are tagged with reals. In our case, we use FADDs with leaves tagged with integers in $[0, m-1]$ denoting actions.

For a factored policy, <plan> is of the form:

```
factored <k> <fadd-elements>
```

where <k> is the size <fadd-elements> which is a space-separated list of variable-sized elements that define the FADD. Each FADD element is either an internal element or a leaf element. An internal element is of the form:

```
I <atom> <left> <right>
```

where <atom> is an integer in $[0, n-1]$ denoting the atom with such index, and <left> and <right> are integers in $[0, k-1]$ denoting the FADD elements with such indexes. The <left> branch corresponds to states when <atom> is true and the <right> branch to states when <atom> is false. A FADD leaf is of the form:

```
L <action>
```

where <action> is an integer in $[0, m]$: if <action> is less than m, it denotes the action with such index, else it denotes the undefined action. Undefined actions are needed in factored representations since not all the valuations of atoms stand for valid states of the problem and/or since the policy doesn't need to be complete in order to be a proper and closed policy. The FADD elements should be listed in inverse topological order of the DAG associated to the FADD.

For example, the file:

```
4 (on A B) (clear A) (clear B) (on B A)
%%
4 (pick A) (putdown A) (pick B) (putdown B)
%%
factored 3
L 0
L 2
I 1 0 1
```

denotes a policy π' such that $S_{\pi'} \supseteq S_\pi$ and that π' agrees with π in S_π, the domain of definition of π, where π' is the policy defined above. The corresponding FADD is depicted in Fig. 1.

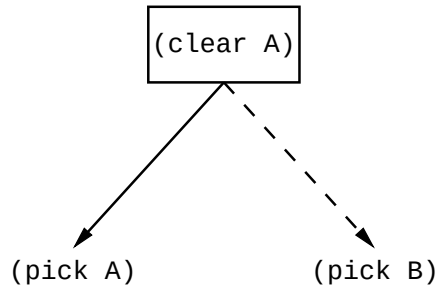

Figure 1: FADD associated with the factored policy in the example text.

Appendix B includes a BNF grammar for the output language used in non-deterministic tracks.

Probabilistic Track The fully observable probabilistic track will use the server-based evaluation of the planning system as was done in IPC-4 and IPC-5. Indeed, the same client-server architecture (MDPSim) will be used, with few changes on the protocol and the evaluation function. In this case, the planner is not required to produce an explicit solution, instead it connects with a server and sends the actions to be executed in a dynamic environment. The planner is evaluated over a number of random 'trials'.

The MDPSim software is available at the following URL: `http://code.google.com/p/mdpsim/` (participants may also want to check `http://www.tempastic.org/mdpsim/`). We make a brief presentation of MDPSim below. More details are available through the README file and the source code.

The `mdpsim` executable is a server used to simulate a variety of probabilistic planning problems specified in PPDDL. It is typically started with a list of PPDDL files as its main argument. In this setting, a planner is viewed as a client software which —when working on a given problem— has 45 minutes to:

1. connect to the server (indicating which domain and problem it wants to work on),
2. search for a policy,
3. interact with the simulator to perform 100 experimental trials[7] and
4. close the connection.

Planning and acting can be interleaved, as a re-planner would do for example. The interaction is a loop alternating between:

- the server simulating the domain and sending the current state to the client, and
- the client making a decision and sending the resulting action to the server.

There are 100 such loops —the 100 experimental trials—, each starting from an initial state and ending in a success state or after a maximum number of iterations (which ensures that all policies are valid).

MDPSim changes once in a while, some bugs being discovered, tracked and fixed, and new features being added. The main novelty in this year's version of MDPSim (more details in the README file) is as follows:

- Up to now, the complete state was sent at each time step, possibly resulting in a very large set of atoms and fluents. To make communications less expensive, a new protocol has been made available, which only transfers state changes.

Partially Observable Probabilistic Track This track still requires a few more entrants before ensuring that it will be organized.

If it is organized, a client-server software will be provided. It's principle should be similar to MDPSim, but on the basis of Cassandra's POMDP file format. As in MDPSim, each planner will be evaluated on each problem by performing 100 experiment trials, each trial ending in a goal state or after a time limit (which ensures that all policies are valid).

[7]The competition makes a move from 30 to 100.

Other Software Planners often have preferences for specific file formats (ADL vs STRIPS) or data structures (matrices vs factored representations). The organizers cannot develop a variety of software tools, but participants are encouraged to share source code (translators, parsers...). The wiki of the uncertainty part of the IPC should be an appropriate place where to store files or URLs.

The non-deterministic plan verifier will also be available on the wiki. The verifier includes a parser as well. However, there is no reference planner implemented for non-deterministic domains.

As a starter, note that MDPSim makes it easy to get C versions of i) a PPDDL parser, ii) a sample planner (`mdpclient`) and iii) a BDD-based planner (`mtbdd`).

Participating

The first thing to do if you want to participate is to contact the organizers by email `ippc-2008-organizers@loria.fr` with the following information:

- Planner name
- Team members and contact information
- Tracks of interests
  - Non-Observable Non-Deterministic
  - Fully-Observable Non-Deterministic
  - Fully-Observable Probabilistic
  - Partially-Observable Probabilistic
- Comments/Questions (schedule, language, etc.)

Note: Up to date deadlines and instructions are available on `http://ipc-2008.loria.fr/wiki/`.

Planner Description

As in previous editions, participants are required to submit a 2–3 page description of their planner. The submission will be in AAAI style. The deadline will be announced at a later date.

Comments

We are open to offering other tracks upon request, if there is sufficient interest and sufficient time available. Another possibility is to leave space for "special entries" (e.g. a planner running on a parallel machine) even if a direct comparison with other competitors is not straightforward. Please inform the organizers of any interests your group has in this or other directions prior to the registration deadline.

References

Bonet, B., and Givan, B. 2005. 5th international planning competition: Non-deterministic track - call for participation. In *Not in the Proceedings of the Fifth International Planning Competition (IPC-5)*.

Bryant, R. 1992. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys* 24(3):293–318.

Goldman, R., and Boddy, M. 1996. Expressive planning and explicit knowledge. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*.

Smith, D., and Weld, D. 1998. Conformant graphplan. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, 889–896.

Younes, H. L. S., and Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University.

Appendix A: (No) BNF for PPDDL

Specifying a complete BNF for the versions of PPDDL that will be used during the competition would be a bit lengthy. A complete BNF for the original PPDDL is available in (Younes & Littman 2004). The following subsections describe the particular versions of PPDDL used in the various tracks (but POP) through changes to the original BNF.

Most of the information below should have already been given earlier in this document.

All Tracks (but POP)

Overview:

Requirements: `:adl` and `:rewards` will be requirements common to all tracks. As a reminder, `:adl` is equivalent to:
`:strips` + `:typing` + `:equality`
+ `:negative-preconditions`
+ `:disjunctive-preconditions`
+ `:quantified-preconditions`
+ `:conditional-effects`

About rewards: Rewards will be the only fluents, the goal being always to maximize the accumulated rewards (whatever the terminal state being reached: the reward will tell if this is a success or failure state).

As already written, the `:adl` requirement will be adopted, i.e. STRIPS with arbitrary conditions and conditional effects, yet no existential quantification, disjunctions or negative literals will be permitted in the preconditions of operators nor in the conditions for conditional effects. However, general formulae will be allowed in the descriptions of the goals. As mentioned in the PPDDL manual, all effects will be order independent and non-conflicting (interfering). *Additionally, in order to ease the development of parsers for PPDDL, all operator schemata will be such that non-determinism inside conditional effects and/or nested conditional effects will not be allowed.*

Both Non-Deterministic Tracks

Requirements: `:non-deterministic` is a new requirement specifying that non-deterministic effects may happen.

Actions: The `probabilistic` statement from PPDDL is replaced by `oneof` (followed by a list of ⟨a-init-el⟩ statements).

Non-Observable Non-Deterministic Track

Requirements: `:non-observable` is a new requirement specifying that no variable is observable. In such a setting, the *reward* is also non observable.

Appendix B: BNF for Output Language

⟨file⟩ ::= ⟨atoms⟩ `%%` ⟨actions⟩ `%%` ⟨plan⟩
⟨atoms⟩ ::= ⟨INT⟩ ⟨ATOM⟩*
⟨actions⟩ ::= ⟨INT⟩ ⟨ACTION⟩*
⟨plan⟩ ::= ⟨linear⟩ | ⟨policy⟩ | ⟨factored⟩
⟨linear⟩ ::= `linear` ⟨INT⟩ ⟨INT⟩*
⟨policy⟩ ::= `policy` ⟨INT⟩ ⟨map⟩*
⟨map⟩ ::= ⟨INT⟩ ⟨INT⟩* ⟨INT⟩
⟨factored⟩ ::= `factored` ⟨INT⟩ ⟨fadd⟩*
⟨fadd⟩ ::= ⟨internal⟩ | ⟨leaf⟩
⟨internal⟩ ::= `I` ⟨INT⟩ ⟨INT⟩ ⟨INT⟩
⟨leaf⟩ ::= `L` ⟨INT⟩