A PDDL+ Benchmark Problem: The Batch Chemical Plant

Giuseppe Della Penna
Dept. of Computer Science
University of L'Aquila
Italy

Benedetto Intrigila
Dept. of Mathematics
University of Rome Tor Vergata
Italy

Daniele Magazzeni
Dept. of Science
University of Chieti
Italy

Fabio Mercorio
Dept. of Computer Science
University of L'Aquila
Italy

Abstract

The PDDL+ language has been mainly devised to allow modelling of real-world systems, with continuous, time-dependant dynamics. Several interesting case studies with these characteristics have been also proposed, to test the language expressiveness and the capabilities of the support tools. However, most of these case studies have not been completely developed so far. In this paper we focus on the *batch chemical plant* case study, a very complex hybrid system with nonlinear dynamics that could represent a challenging benchmark problem for planning techniques and tools. We present a complete PDDL+ model for such system, and show an example application where the UPMurphi universal planner is used to generate a set of *production policies* for the plant.

1. Introduction

The batch chemical plant is an academic case study, originally developed in (Kowalewski 1998), describing a chemical production cycle for concentrated saline solution. Its aim is to model a realistic hybrid system to experiment various model checking and planning techniques and tools. The system has been also studied in-depth as the subject of the European project VHS (Verimag 2000).

It has been remarked (Fox, Howey, and Long 2005; Fox and Long 2006) that the chemical batch plant could be an interesting case study to fully exploit the modelling power of PDDL+ (Fox and Long 2001). Therefore, it could also represent a challenging benchmark for all the current planning tools that support this language. Indeed, the problem domain contains many tightly connected components and safety constraints, which make the actual effects of any action very difficult to estimate.

The general problem of synthesising a set of operating procedures (plan) for a chemical plant has been addressed several times in the literature (see, e.g., (Ivanov et al. 1980)). Typically, the problem solution involves the use of hierarchical non-linear planners to generate partial plans and refine them until a feasible global plan is generated, following the plant *flow*. Examples of such approach are, e.g., (Aylett et al. 1998), where the CEP domain-dependent planner is applied to a chemical processing case study, or (Viswanathan et al. 1998a), where the Grafchart methodology is used in the *i*TOPS (Viswanathan et al. 1998b) tool for synthesising a detailed plan for a batch chemical process, or (Crooks and Macchietto 1992), where the goal of each subplan is solved as a mixed integer linear problem. However, the major issue of all these refinement-based techniques is that, when the model presents a large number of constraints, detecting their violation, the invalidated subplans and the corresponding possible corrections may quickly become a very hard task.

Therefore, so far a complete batch chemical plant like the one described in (Kowalewski 1998) has not been modelled in PDDL+, neither an automatic planning tool (possibly using an alternative input representation) has been applied to it.

In this paper we propose a complete PDDL+ model for the batch chemical plant. Such model represents an interesting and advanced application of PDDL+, and we feel that it could also be a challenging benchmark problem for planning and universal planning tools. Indeed, we also describe a sample case study using the UPMurphi (Della Penna et al. 2009b) tool to perform universal planning (Schoppers 1987) on a significant portion of the model, in order to generate a set of *production policies* for the plant.

2. The Batch Chemical Plant

The purpose of the batch chemical plant is to produce saline solution with a given concentration. If part of the product is not used, the plant can recycle it to restart another production cycle.

The plant (shown in Figure 1) is composed of 7 tanks connected through a complex pipeline, whose flow is regulated by 26 valves and two pumps. In particular, tank 5 is provided with a heater, whereas tank 6 is connected to a condenser. Finally, tanks 6 and 7 are surrounded by a cooling circuit. A set of sensors provide information to the plant controller about the filling level of tanks 1,2,3 and 5, the pump pressure and the condenser status.

In the plant initial state, all the valves are closed, and the pumps, heaters and coolers are switched off. Tank 1 contains saline solution at a high concentration c_{high}, whereas tank 2 contains water.

If tank 1 does not contain enough solution, the plant enters the *startup phase*: water from tank 2 is moved to tank 3, where a suitable amount of salt is added manually to reach

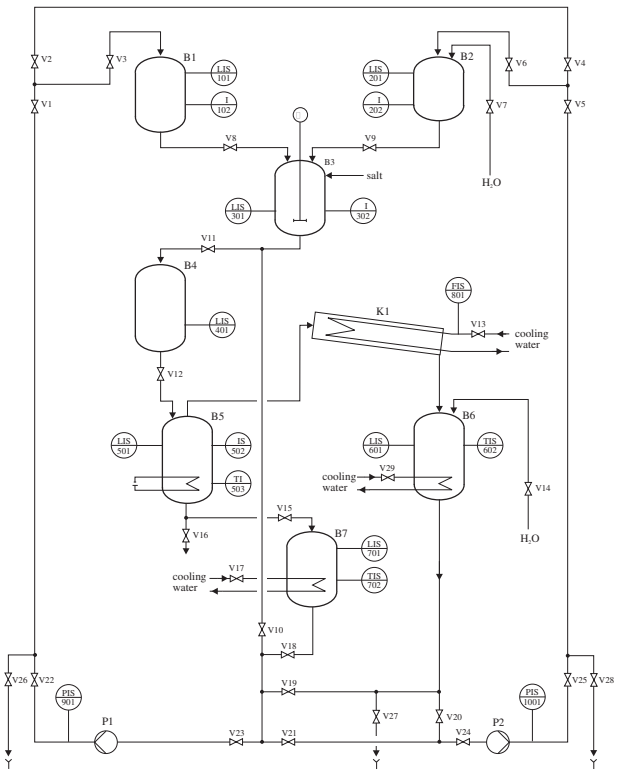

Figure 1: Overall structure of the chemical plant

the required concentration, and finally pumped to tank 1. Note that tank 2 can be refilled with water at any time by opening the appropriate input valve.

When tanks 1 and 2 are appropriately filled, the plant can start the *production phase*. Tank 3 is partially filled with the solution from tank 1, which is then diluted using the water from tank 2 up to the requested concentration.

The resulting saline solution can be taken from the output valve of tank 3. If the product is not completely used, the plant recycles it in the next production cycle. To this aim, the solution in tank 3 is moved to tank 4 and then to tank 5. Here, the solution is boiled by the heater until it reaches the concentration c_{high}, and then moved to tank 7. The steam produced by this process is piped to the condenser that fills tank 6 with the resulting water. Finally, tanks 6 and 7 are cooled and their contents are pumped to tanks 2 and 1, respectively.

During the startup and production cycles the plant must obey some safety rules:

- pumps can be switched on only if all the valves in their pipeline are open,
- the heater cannot be switched on if tank 5 is empty, or the condenser is switched off, or if the valves involved in the heating/condensation process are closed,
- only two cooling circuits (including the one used by the condenser) can be switched on at the same time,
- tanks cannot be filled and emptied at the same time,
- the content of each tank must not exceed the corresponding capacity limitations (Kowalewski 1998), which are lower than the tank volume.

The plant dynamics is described in (Deparade 1999) by a set of differential equations that we omit for lack of space.

3. PDDL+ Modelling

In this paper, for sake of simplicity, we only describe the (more interesting) model of the production phase. This continuous, time-dependant domain is mainly modelled using processes, events and (flexible) durative actions. Indeed, Figures 2, 4 and 5 show representative examples of such constructs extracted from the model (whose full source is available online in (Della Penna et al. 2009a)), which contains a total of 59 predicates, 55 functions (14 of which represent real values), 19 events, 10 durative actions and 11 processes. In the figures, Bx_l, Bx_c, Bx_t indicate the filling level, solution concentration and temperature for tank x, respectively, whereas Vy, Py and Hy indicate valve, pump and heater y, respectively. Finally, the value of a constant k taken from the problem specification is indicated with C_k.

In the following we describe the main elements of the PDDL+ model for the chemical plant production phase, highlighting their most interesting features. It is worth noting that the model has been written to adhere as much as possible to the formal specification in (Deparade 1999). However, to further check its correctness, we used the UPMurphi tool (Della Penna et al. 2009b) to generate an optimal production plan, using our model applied to the initial conditions described in (Kowalewski 1998) and obtaining the same results (manually) devised in (Kowalewski 1998).

3.1 Production Activities

```
; filling durative action (for tank 3)
(:durative-action B3_fill
:parameters () :duration (>= ?duration 0)
:condition (and
 (at start (not (V8))) (at start (= (B3_l) 0))
 (at start (>= (B1_l) 0)) (at start (not (V3)))
 (at start (not (V10))) (at start (not (V11)))
 (at start (not (B3_filled))) (at end (V8))
 (over all (>=(B1_l) 0)))
:effect (and
 (at start (B3_filling)) (at start (V8))
 (at end (not (V8))) (at end (B3_filled))
 (at end (not (B3_filling)))))
; filling process (for tank 3)
(:process B3_fill_process
:parameters ()
:precondition (B3_filling)
:effect (and (decrease (B1_l) (* #t (* (C_5_2) (
 sqrt (+ (/ (B1_l) (C_h_1_3)) 1 )))))
 (increase (B3_l) (* #t (* (C_5_2) (sqrt (+ (/ (
  B1_l) (C_h_1_3)) 1 )))))))
```

Figure 2: Examples of durative actions and processes modelling the production phase

The production activities, such as moving the solution from a tank to another, cool it down, etc., some of which can possibly be executed in parallel, are modelled using durative actions. However, the duration of these activities is not known *a priori*, thus the planner should determine the time point at which the tank capacity (or required concentration, or temperature) is reached. To achieve this, we use *duration inequalities* in the durative actions. On the other hand, continuous changes to solution level, concentration and temperature in tanks are modelled through PDDL+ processes that update the corresponding model variables following the functions described in (Deparade 1999). This modelling schema guarantees an immediate detection (i.e., triggering of failure events) of safety violations.

As an example, when tank 1 is nonempty, tank 3 is empty and some other conditions hold, the durative action `B3_fill` shown in Figure 2 moves the solution from tank 1 to tank 3. The continuous update to the solution level in these tanks due to the action is performed by the process `B3_fill_process`, which is enabled by the durative action by setting to true the predicate `B3_filling`. The execution of this process may in turn trigger some events (Fox and Long 2003), e.g., `B3_l_failure` (shown in Figure 4) that would invalidate the plan. At the end of the durative action (as chosen by the planner), `B3_filling` is set to false, and the filling process ends.

It is worth noting that the effects of `B3_fill_process` involve the calculation of a square root, which is currently not supported by PDDL+. Therefore, we have also created and tested an *approximated* model (available in (Della Penna et al. 2009a)), where the square root is substituted by the second degree polynomial that best fits such function within the bounds deducible from the model dynamics, as shown in Figure 3.

```
(:process B3_fill_process
:parameters ()
:precondition (B3_filling)
:effect (and (decrease (B1_l) (* #t (* (C_5_2)
  (+ (* -0.000415797 (* (B1_l) (B1_l) ) ) (+ (* (
B1_l) 0.0424115 ) 1.00597 )))))
 (increase (B3_l) (* #t (* (C_5_2)
  (+ (* -0.000415797 (* (B1_l) (B1_l) ) ) (+ (* (
 B1_l) 0.0424115 ) 1.00597 )))))))
```

Figure 3: `B3_fill_process` with approximated square root

3.2 Production Events

The violation of one of the safety constraints listed in Section 2 should trigger an instantaneous change that invalidates the plan. Therefore, such failures have been modelled through PDDL+ events, whose effect is to falsify the invariant predicate `correct_operation`.

It is worth noting that, in the chemical plant model, discrete and continuous changes are combined in the activation conditions of several events (Howey, Long, and Fox 2004), making their checking more complex, but still very important since they may invalidate the plan (Fox, Howey, and Long 2005). As an example, event `H5_failure` in Figure 4 shows the PDDL+ model of an exogenous event. Such event is activated when the heater is switched on (`H5` is true) and one of the valves 12, 15 or 16 is open (*or* `V12 V15 V16`), or valve 13 is closed (*not* `V3`), or the level of tank 5 is lower than the security level (*not* `(>= B5_l B5_l_safe)`).

```
; pipeline flow failure (during B3 filling process)
(:event B3_flow_failure
:parameters ()
:precondition (and (or (V11) (V10)) (or (V8) (V9)))
:effect (not (correct_operation)))
; heater failure (on tank 5)
(:event H5_failure
:parameters ()
:precondition (or (and (H5) ( or (V12) (V15) (V16 )
)) (and (H5) (not(V13))) (and (H5) (not (>= (B5_l)
 (B5_l_safe)))))
:effect (not (correct_operation)))
; tank filling limit failure (on tank 3)
(:event B3_l_failure
:parameters ()
:precondition (or (< (B3_l) 0) (> (B3_l) (B3_l_max)
))
:effect (not (correct_operation)))
; pump (2) failure
(:event P2_failure
:parameters ()
:precondition (and (P2) (not(or (and (V25) (V28)) (
and (V25) (V5) (V6)) (and (V25) (V5) (V4) (V2) (V1
) (V3)))))
:effect (not (correct_operation)))
```

Figure 4: Examples of failure events

Finally, the two events shown in Figure 5 are used to trigger the end of the plan. In particular, event `production_end` is triggered when tank 1 contains a sufficient amount of solution with the required concentration, and its effect is to set the `production_complete` predicate to true. This, in turn, triggers a *cascading* event `production_success` that, if the plant has operated correctly (i.e., without violating any safety constraint) and all the valves and pumps have been correctly closed, sets the `success` predicate to true to indicate that the goal has been reached.

3.3 Production Problem

The PDDL+ definition of the problem for the chemical plant production phase is quite straightforward. The domain is initialised by setting the function and predicate values to the ones obtained after the startup phase (see (Deparade 1999)), and the goal is to set the `success` predicate to true, minimising the `total-time`.

4. Experimentation

As a first application of the chemical plant model, we used the UPMurphi tool (Della Penna et al. 2009b) to automatically perform universal planning (Schoppers 1987) on the production phase, in order to generate a set of *production*

```
(:event production_end
:parameters ()
:precondition (and
 (B1_filled) (>= (B1_l) (B1_l_target_min))
 (< (B1_l) (B1_l_target_max))
 (= (B1_c) (B1_c_target)) (not (production_ended)))
:effect (and (production_complete)
 (production_ended)))
(:event production_success
:parameters ()
:precondition (and (not (success))
 (production_complete) (correct_operation)
 (not (or (V1) (V2) (V3) (V4) (V5) (V6) (V7) (V8) (
 V9) (V10) (V11) (V12) (V13) (V14) (V15) (V16) (
 V17) (V18) (V19) (V20) (V21) (V22) (V23) (V24) (
 V25) (V26) (V27) (V28) (V29) (P1) (P2))))
:effect (success))
```

Figure 5: Cascading events triggering the goal

policies with different requirements. To this aim, we input UPMurphi with the PDDL+ domain and problem, and define a *start state cloud* (see (Della Penna et al. 2009b)) where the amount of solution to be produced (`B3_l_target`) varies in the range $[1.5, 3.7]$ liters with steps of 0.1 (i.e., 23 different states).

Note that, to obtain the finite state system needed by UPMurphi, the continuous variables of the model have been rounded up to the first decimal, and the time has been discretised in steps of 10 seconds, as suggested in (Brinksma and Mader 2000).

The whole universal plan took about 6000 seconds to be generated, and has a size of 246.3 megabytes. The results are in Table 1. The plant state space is 10^{17}. However, starting from the given *start state cloud*, the planner found that only about 30 million of such states were actually reachable, and for 24% of them it was able to generate an optimal plan to the goal. Plan execution lengths vary from 1920 to 2200 seconds.

Reachable states	$29,968,861$
States to goal (generated plans)	$7,154,464$
Optimal plan length (min/max)	1920/2200 sec

Table 1: Universal plan generation statistics

5. Conclusions

The complete PDDL+ model for the batch chemical plant is now available to the planning community, together with the preliminary results of the universal planning presented in this paper. As a complex benchmark problem, it can be used to validate PDDL+ tools, but also to highlight flaws and suggest possible enhancements.

References

Aylett, R.; Soutter, J. K.; Petley, G. J.; and Chung, P. W. H. 1998. AI planning in a chemical plant domain. In *Proc. ECAI 1998*, 622–626.

Brinksma, E., and Mader, A. 2000. Verification and optimization of a PLC control schedule. In *Proc. SPIN 2000*, 73–92.

Crooks, C., and Macchietto, S. 1992. A combined MILP and logic-based approach to the synthesis of operating procedures for batch plants. *Chem. Eng. Comm.* 114:117–144.

Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2009a. Batch chemical plant PDDL+ model. `http://www.di.univaq.it/gdellape/lamoka/go/?page=chemical`.

Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2009b. UPMurphi: a tool for universal planning on PDDL+ problems. In *Proc. ICAPS 2009*, 106–113. AAAI Press.

Deparade, A. 1999. A switched continuous model of VHS case study 1. Draft, University of Dortmund, `http://www-verimag.imag.fr/VHS/year1/cs11c.ps`.

Fox, M., and Long, D. 2001. PDDL+: An extension to PDDL2.1 for modelling planning domains with continuous time-dependent effects. *Technical Report, Dept. of Computer Science, University of Durham*.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* 20:61–124.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.* 27:235–297.

Fox, M.; Howey, R.; and Long, D. 2005. Validating plans in the context of processes and exogenous events. In *Proc. IAAI 2005*, 1151–1156. AAAI Press.

Howey, R.; Long, D.; and Fox, M. 2004. Validating plans with exogenous events. In *Proc. 23rd Workshop of the UK Planning and Scheduling Special Interest Group*.

Ivanov, V. A.; Kafarov, V. V.; Perov, V. L.; and Reznichenko, A. A. 1980. On algorithmization of the start-up of chemical productions. *Engineering Cybernetics* 18:104–110.

Kowalewski, S. 1998. Description of VHS case study 1 "Experimental Batch Plant". Draft. University of Dortmund, Germany `http://astwww.chemietechnik.uni-dortmund.de/~vhs/cs1descr.zip`.

Schoppers, M. 1987. Universal plans of reactive robots in unpredictable environments. In *Proc. IJCAI 1987*.

Verimag. 2000. ESPRIT-LTR project 26270 (verification of hybrid systems). `http://www-verimag.imag.fr/VHS/`.

Viswanathan, S.; Johnsson, C.; Srinivasan, R.; Venkatasubramanian, V.; and Arzen, K. E. 1998a. Automating operating procedure synthesis for batch processes: Part I. Knowledge representation and planning framework. *Computers and Chemical Engineering* 22(11):1673–1685.

Viswanathan, S.; Mockus, L.; Venkatasubramanian, V.; Basu, P. K.; Mack, R.; Cherukat, P.; and Iskos, V. 1998b. iTOPS: An intelligent tool for operating procedures synthesis. *Computers and Chemical Engineering* 22(Supplement 1):S601 – S608.