A Scheduling Problem

Assaf Kfoury

September 16, 2020

There are six tasks each with an associated duration to be completed:

$$\{A : 2,\ B : 1,\ C : 2,\ D : 2,\ E : 7,\ F : 5\}.$$

The intended meaning is that task A requires two time units to complete, task B requires one time units to complete, etc. We impose scheduling constraints on the order in which these tasks should be completed:

- Tasks A and C may not overlap.
- No two of tasks B, D or E can overlap.
- Tasks D and E must be completed before task F starts.
- Task A must complete before B starts.

We can introduce integer variables $\{X, X_t\}$ where X holds the starting time of task X starts and X_t holds the time required for X to complete. The overlapping constraints are easy to encode. If task A does not overlap with task C, then one of two conditions must be true:

1. Task A must complete before task C,
2. Task C must complete before task A.

We express the preceding two conditions as a disjunction:

$$(A + A_t \leqslant C)\ \vee\ (C + C_t \leqslant A)$$

Similarly, all the other constraints can be expressed by the following wff's:

$$(B + B_t \leqslant D)\ \vee\ (D + D_t \leqslant B) \qquad (B \text{ and } D \text{ do not overlap})$$

$$(B + B_t \leqslant E)\ \vee\ (E + E_t \leqslant B) \qquad (B \text{ and } E \text{ do not overlap})$$

$$(D + D_t \leqslant E)\ \vee\ (E + E_t \leqslant D) \qquad (D \text{ and } E \text{ do not overlap})$$

$$(D + D_t \leqslant F)\ \wedge\ (E + E_t \leqslant F) \qquad (D \text{ and } E \text{ must complete before } F)$$

$$(A + A_t \leqslant B) \qquad (A \text{ must complete before } B)$$

In addition to the preceding requirements, we want the start time of task X to be a non-negative integer. This requires the following constraints:

$$A \geqslant 0, \quad B \geqslant 0, \quad C \geqslant 0, \quad D \geqslant 0, \quad E \geqslant 0, \quad F \geqslant 0.$$

Moreover, if End is a time limit for all the tasks to be completed, we also have the constraints:

$$A + A_t \leqslant \mathsf{End}, \quad B + B_t \leqslant \mathsf{End}, \quad C + C_t \leqslant \mathsf{End}, \quad D + D_t \leqslant \mathsf{End}, \quad E + E_t \leqslant \mathsf{End}, \quad F + F_t \leqslant \mathsf{End}.$$

Using the notational conventions of Z3, all of the preceding constraints are shown in Figure 1.

We are here using Z3 as a SMT solver, not just as a SAT solver, because we are supplying it with a background theory, namely, the *quantifier-free linear integer arithmetic* (`QF_LIA`); this is indicated by the first line at the top of the script in Figure 1. The second line asks Z3 to keep the model it generates for subsequent printing.

If this script is saved in a file called `schedule.smt2`, issuing the command "`z3 schedule.smt2`" at the Unix/Linux prompt should return the following:

```
sat
((A 0)
 (B 9)
 (C 2)
 (D 0)
 (E 2)
 (F 9))
```

Caution: For some reason, the last command in my script, `(get-value (A B C D E F))`, causes an error. However, if we break up the command into three consecutive commands:

```
(get-value (A B)) (get-value (C D)) (get-value (E F))
```

the error disappears! I have no idea why this is happening.

```
(set-logic QF_LIA)
(set-option :produce-models true)

(declare-fun A () Int)
(assert (>= A 0))
(declare-fun At () Int)
(assert (= At 2))
(declare-fun B () Int)
(assert (>= B 0))
(declare-fun Bt () Int)
(assert (= Bt 1))
(declare-fun C () Int)
(assert (>= C 0))
(declare-fun Ct () Int)
(assert (= Ct 2))
(declare-fun D () Int)
(assert (>= D 0))
(declare-fun Dt () Int)
(assert (= Dt 2))
(declare-fun E () Int)
(assert (>= E 0))
(declare-fun Et () Int)
(assert (= Et 7))
(declare-fun F () Int)
(assert (>= F 0))
(declare-fun Ft () Int)
(assert (= Ft 5))

(assert (or (<= (+ A At) C) (<= (+ C Ct) A)))
(assert (or (<= (+ B Bt) D) (<= (+ D Dt) B)))
(assert (or (<= (+ B Bt) E) (<= (+ E Et) B)))
(assert (or (<= (+ D Dt) E) (<= (+ E Et) D)))
(assert (and (<= (+ D Dt) F) (<= (+ E Et) F)))
(assert (<= (+ A At) B))

(declare-fun End () Int)
(assert (= End 14))
(assert (<= (+ A At) End))
(assert (<= (+ B Bt) End))
(assert (<= (+ C Ct) End))
(assert (<= (+ D Dt) End))
(assert (<= (+ E Et) End))
(assert (<= (+ F Ft) End))

(check-sat)
; (get-model)    ; un-comment if you want to see details of the model
(get-value (A B C D E F))
```

Figure 1: Encoding the scheduling problem in the conventions of Z3.