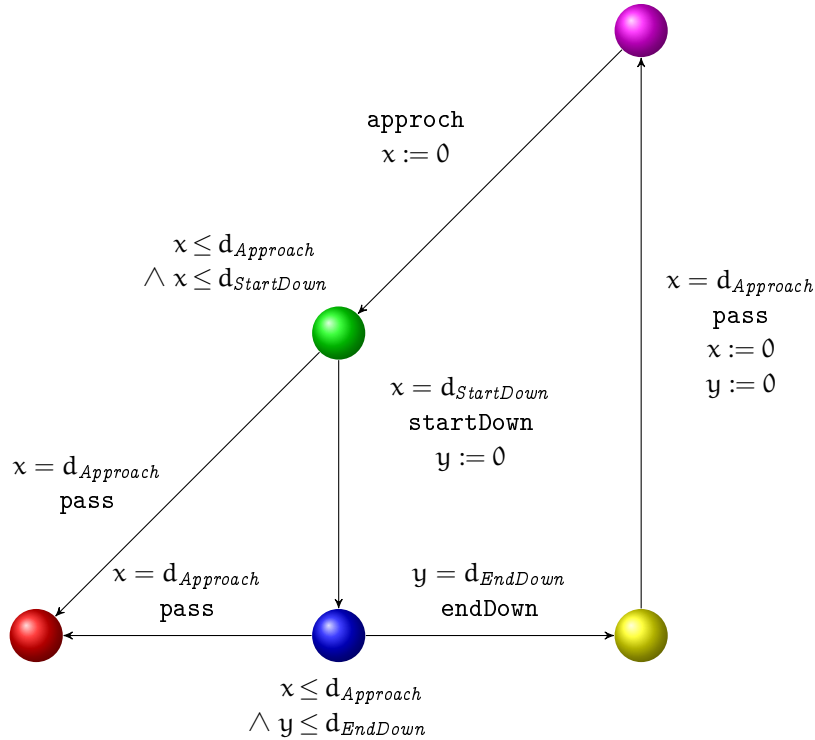


Soit un passage à niveau dont le modèle est représenté ci-dessous sous la forme d'un automate temporisé [AD94]. Un automate temporisé est un automate à états finis, avec des horloges (ci-dessous  $x$  et  $y$ ), c'est-à-dire des variables dont la valeur augmente linéairement à la même vitesse. Ces variables peuvent être réinitialisées sur des transitions ( $x := 0$ ), être comparées à des constantes dans des gardes en franchissant une transition ( $x = d_{Approach}$ ) ou dans des invariants afin de rester dans un état de contrôle ( $x \leq d_{Approach}$ ).

Dans ce modèle, lorsqu'un train s'approche (événement *approach*), un signal est émis vers le passage à niveau. On sait alors que l'on dispose d'exactly  $d_{Approach}$  unités de temps avant que le train n'arrive sur le passage à niveau (événement *pass*). On attend alors  $d_{StartDown}$  unités de temps avant de commencer à faire descendre la barrière (événement *startDown*). Celle-ci met  $d_{EndDown}$  unités de temps à se fermer (événement *endDown*).


Le but est bien entendu que, lorsque le train arrive, la barrière soit fermée, sinon c'est le crash.

L'automate temporisé paramétré suivant modélise ce comportement ;  $x$  et  $y$  sont deux horloges. Le système commence dans l'état de contrôle  $\bullet$ , avec  $x = y = 0$ .



Notons que, dans l'état  $\bullet$ , le train passe quand la barrière est bien fermée alors que, dans l'état  $\bullet$ , il passe alors que la barrière n'est pas (entièrement) fermée.

Les 3 valeurs  $d_{Approach}$ ,  $d_{StartDown}$  et  $d_{EndDown}$  sont appelées des *paramètres* : ce sont des constantes, qui ne seront donc pas modifiées pendant l'exécution, mais dont la valeur est inconnue. Ce type de modèle est donc un automate temporisé paramétré [AHV93]. Nous allons dans la suite

déterminer des valeurs des 3 paramètres  $d_{Approach}$ ,  $d_{StartDown}$  et  $d_{EndDown}$  permettant de garantir que, pour toute exécution, le système ne pourra pas entrer dans l'état .

### Exercice 1 : Synthèse de paramètres avec la méthode inverse

À partir d'une valeur connue des paramètres, la méthode inverse [AS13] généralise cette valeur des paramètres sous la forme d'une contrainte entre les paramètres. Pour toute valeur des paramètres dans cette contrainte, le comportement (ensemble de traces) est le même que pour la valeur de départ.

**Question 1 :** Soit la valeur des paramètres suivante :  $d_{Approach} = 5 \wedge d_{StartDown} = 2 \wedge d_{EndDown} = 2$ . Quelle est la contrainte généralisant cette valeur ? Concrètement dans *CosyVerif*, il faut choisir le service « méthode inverse », et entrer la valeur de référence suivante :

```
& dApproach = 5
& dStartDown = 2
& dGetDown = 2
```

La contrainte résultat est donnée dans la fenêtre de résultat, après « Final constraint K0 ».

On peut également voir l'ensemble de traces correspondant (en cliquant sur « next »). Est-ce que le système peut atteindre l'état crash ?

**Question 2 :** Mêmes questions avec  $d_{Approach} = 2 \wedge d_{StartDown} = 5 \wedge d_{EndDown} = 2$ .

## Références

- [AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, April 1994.
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC'93, pages 592–601, New York, NY, USA, 1993. ACM.
- [AS13] Étienne André and Romain Soulat. *The Inverse Method*. FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc., 2013. 176 pages.