Vérification formelle de conditions d'ordonnançabilité de tâches temps réel périodiques strictes

Daniel de Rauglaudre

Équipe-projet Aoste - Inria

6 février 2012

Plan

Plan

Introduction

Motivation :

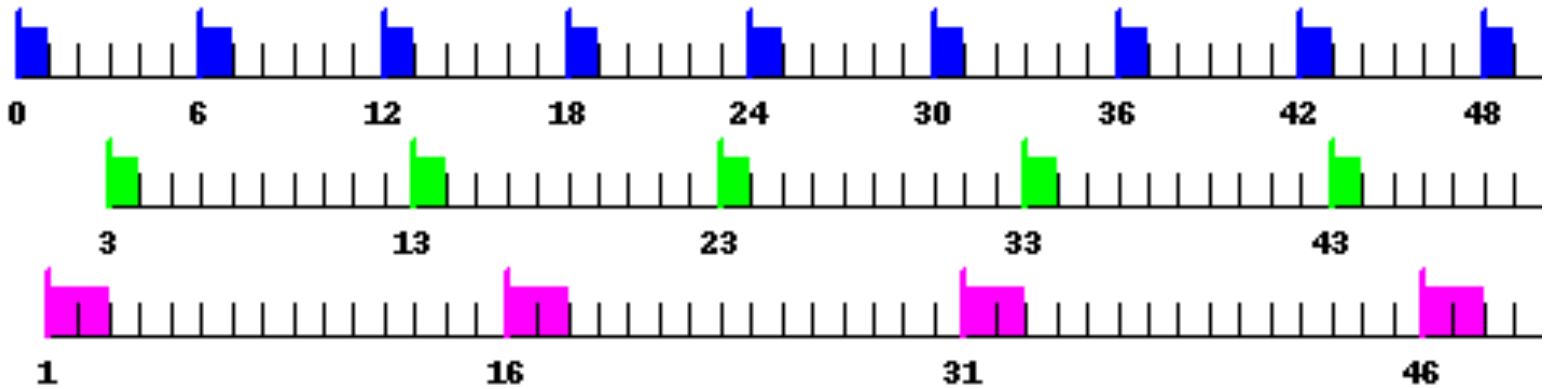
- Systèmes embarqués : sécurité.
- Tester preuves formelles.
- Projet P (Oséo).

Logiciels d'ordonnancement :

- SynDEx de l'équipe Aoste (multiprocesseur).
- Dameid.

Exemple de trois tâches

tâche τ_1 : $\{\text{phase} = 0\,;\ \text{durée} = 1\,;\ \text{période} = 6\}$
tâche τ_2 : $\{\text{phase} = 3\,;\ \text{durée} = 1\,;\ \text{période} = 10\}$
tâche τ_3 : $\{\text{phase} = 1\,;\ \text{durée} = 2\,;\ \text{période} = 15\}$



Plan

Théorème (Jan Korst - 1992)

Une liste de tâches $\{\tau_1 .. \tau_n\}$ est ordonnançable par l'algorithme de *périodicité stricte non préemptive*, si et seulement si :

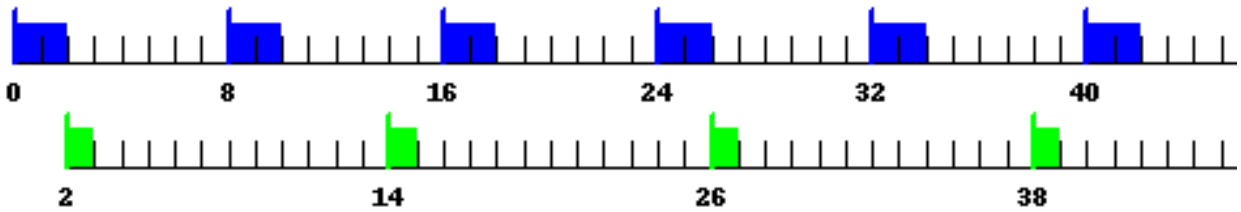
$$\forall\ \tau_i = (\phi_i,\ d_i,\ P_i) \in \{\tau_1 .. \tau_n\}$$
$$\forall\ \tau_j = (\phi_j,\ d_j,\ P_j) \in \{\tau_1 .. \tau_n\}$$
$$\text{avec i} \neq \text{j et } \phi_i \leq \phi_j$$

on ait :

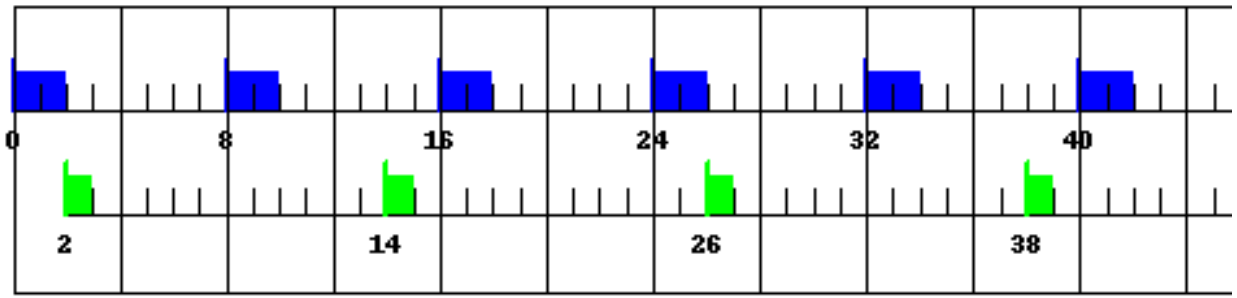
$$d_i \leq (\phi_j \text{ - } \phi_i) \bmod G_{i,j} \leq G_{i,j} \text{ - } d_j$$

où $G_{i,j} = \text{pgcd}\ (P_i,\ P_j)$

Représentation de $d_i + d_j \leq G_{i,j}$



le pgcd des périodes vaut 4 :



Plan

Structures de données + Exemple

Tâche :

```
Record Task :=
  { id : nat ; phase : nat ; duration : nat ; period : nat }.
```

Unité de temps :

```
Inductive time_unit :=
 | E : time_unit                (* processeur libre *)
 | A : nat → time_unit.         (* processeur occupé *)
```

Exemple :

A 1, A 3, A 3, A 2, E, E, A 1, E, E, E, E, E, A 1, ...

Fonctions

nth_time_unit : nat → Task → time_unit

- renvoit la t^{ieme} unité de temps pour la tâche τ

merge_unit_list : nat → list Task → option time_unit

- renvoit la t^{ieme} unité de temps pour la liste de tâches τl

schedulable : list Task → Prop :=

- $\forall t$, merge_unit_list t τl $\neq$ None

Condition nécessaire

Si, pour tout couple de tâches (τ_i, τ_j) de τl,

$d_i \leq (\phi_j - \phi_i) \bmod G_{i,j} \leq G_{i,j} - d_j$

alors

$\forall$ t, merge_unit_list t τl $\neq$ None.

Preuve :

Par contradiction. On suppose donc que :

$\exists\ \tau_i\ \tau_j$ t / nth_time_unit t τ_i = A i
nth_time_unit t τ_j = A j

Puis, considérations sur les opérations élémentaires, les inégalités et les modulos. Voir article.

Réciproque

On doit montrer que si

$\forall$ t, merge_unit_list t τI $\neq$ None,

alors, pour tout couple de tâches (τ_i, τ_j) de τI,

$d_i \leq (\phi_j$ - $\phi_i)$ mod $G_{i,j} \leq G_{i,j}$ - d_j

Preuve : par la contraposée.

mais... mais... est-ce que ça va résoudre le problème ?

Contraposée... problème !

En logique intuitioniste :

$$(A \implies B) \implies (\bar{B} \implies \bar{A})$$

Mais la réciproque n'est pas vraie, en général.

Sauf si B est décidable !

B =

Pour tout couple de tâches (τ_i, τ_j) de τI,

$$d_i \leq (\phi_j - \phi_i) \bmod G_{i,j} \leq G_{i,j} - d_j$$

Contraposée de la réciproque

S'il existe un couple de tâches (τ_i, τ_j) de τl tel que :

$d_i > (\phi_j - \phi_i) \bmod G_{i,j} \vee$

$(\phi_j - \phi_i) \bmod G_{i,j} > G_{i,j} - d_j$

alors

$\exists$ t, merge_unit_list t τl = None.

Preuve : théorème de Bachet-Bézout.

Contraposée de la réciproque

S'il existe un couple de tâches (τ_i, τ_j) de τl tel que :

$d_i > (\phi_j - \phi_i) \bmod G_{i,j} \vee$
$(\phi_j - \phi_i) \bmod G_{i,j} > G_{i,j} - d_j$

alors

$\exists$ t, merge_unit_list t τl $=$ None.

Preuve : théorème de Bachet-Bézout.

$\forall$ a b $\in \mathbb{N}$, $\exists$ u v $\in \mathbb{Z}$ / a.u + b.v $=$ pgcd(a, b)

Contraposée de la réciproque

S'il existe un couple de tâches (τ_i, τ_j) de τl tel que :

$d_i > (\phi_j - \phi_i) \bmod G_{i,j} \vee$
$(\phi_j - \phi_i) \bmod G_{i,j} > G_{i,j} - d_j$

alors

$\exists$ t, merge_unit_list t τl = None.

Preuve : théorème de Bachet-Bézout.

$$\forall\ a\ b \in \mathbb{N}, \exists\ u\ v \in \mathbb{Z}\ /\ a.u + b.v = pgcd(a, b)$$

Bachet-Bézout sur les entiers naturels :

$$\forall\ a\ b \in \mathbb{N}, \exists\ u\ v \in \mathbb{N}\ /\ a.u = b.v + pgcd(a, b)$$

Corollaire 1 : somme des durées

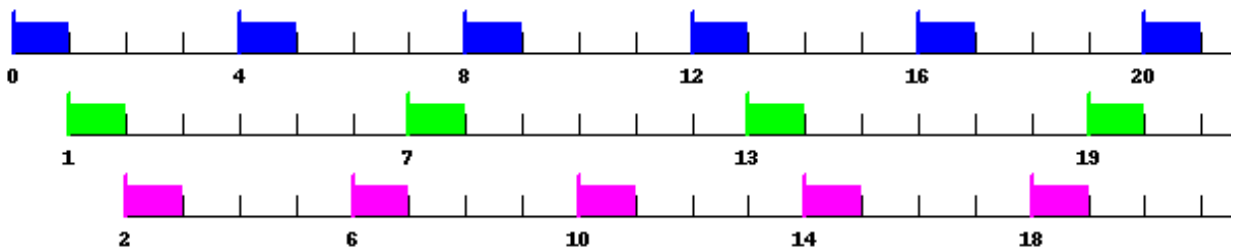
Une liste de tâches $\{\tau_1 \,..\, \tau_n\}$ est ordonnançable si :

$$\sum_{\tau_k} d_k \leq \text{pgcd } P_k$$

On choisit les phases : $\phi_1 = 0$ et $\phi_{i+1} = \phi_i + d_i$

Réciproque fausse en toute généralité. Exemple :

$$\{d_1{=}1\,;\, P_1{=}4\,\}\ \{d_2{=}1\,;\, P_2{=}6\,\}\ \{d_3{=}1\,;\, P_3{=}4\,\}$$



Corollaire 2 : deux tâches

Deux tâches τ_1 et τ_2 sont ordonnançables si et seulement si :

$$d_1 + d_2 \leq \text{pgcd}(P_1, P_2)$$

Phases : $\phi_1 = 0$ et $\phi_2 = d_1$

Réciproque vraie dans ce cas.

Plan

Concrètement...

- début 2011
- 2 mois de développement (débutant Coq)
- 2000 lignes de Coq
- sources sur le Web :
  `http://pauillac.inria.fr/~ddr/publi/coq_korst/`
- travail clos

Conclusion et Travaux futurs

- Preuves *formelles* possibles dans ce domaine.
- Opinion perso : Coq = révolution dans les maths !
- Autre théorème en cours : tâches préemptives avec priorité.
- Preuves formelles de *programmes* embarqués ? Projet P (Oséo).
- Preuve formelle du noyau de SynDEx ?
- Tactiques : *ssreflect* de Microsoft Research-Inria ?