Journée de l'Institut Farman

ENS Cachan

26 novembre 2013

Projet ROSCOV
Robuste Ordonnancement de Systèmes de Contrôle de Vol

L. Fribourg, D. Lesens, P. Moro, R. Soulat

Le projet ROSCOV

Collaboration entre un laboratoire et un partenaire industriel

- LSV (Laboratoire Spécification et Vérification)
  - Étienne André
  - Laurent Fribourg
  - Giuseppe Lipari
  - Romain Soulat
  - Youcheng Sun
- Astrium Space Transportation
  - David Lesens
  - Pierre Moro

Historique : accord-cadre ENSC-Astrium

- Rencontre Christian-Florian-Laurent Astrium, Les Mureaux 2011

- Initiation de coopérations thématiques
- Challenge 1 : participer à l'analyse d'un projet d'ordonnanceur pour le logiciel de vol de la prochaine génération de lanceur (Ariane 6, ATV,...)

Cadre

- état de l'art (Ariane 5) : architecture centralisée, tâches principales exécutées sur le même processeur
-
- D'autres équipements avec des functions intelligentes (recueil des mesures des capteurs,...) sont reliés en mode maître/esclave au processeur central
- $\rightsquigarrow$ determinisme

Nouvelle architecture

- Idée héritée de l'A380 Avionics

- Integrated Modular Avionics : Au lieu d'1 processeur/contrôleur pour chaque (sous-)système, intégration de différentes applications sur un même ordinateur avec un bus de communication reliant les les différents ordinateurs
- avantages : modularité, gain en poids
- difficulté : nécessité d'un ordonnancement distribué des tâches plutôt que centralisé

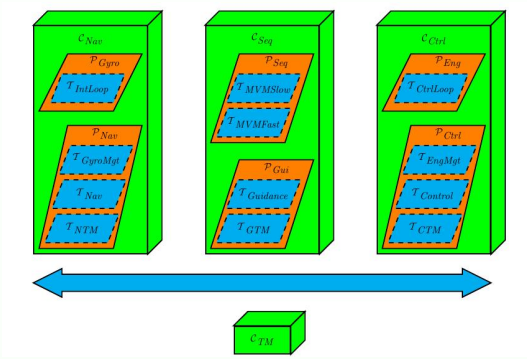
Nouvelle architecture (2)

- Avionic-X (Next Generation Launcher avionics) : cut cost or die

- intégration d'équipements complexes (e.g., SRI : Inertial Measurement Unit et OBC : On-Board Computer) $\rightsquigarrow$ poids $\downarrow$
- impact sur l' architecture :
  information à collecter et répartir vers différents équipements
  - nouveau mode de séparation entre parties de logiciel : partitionnement
  - nouveau mode de communication : bus interne à haut débit

Etude de cas ASTRIUM

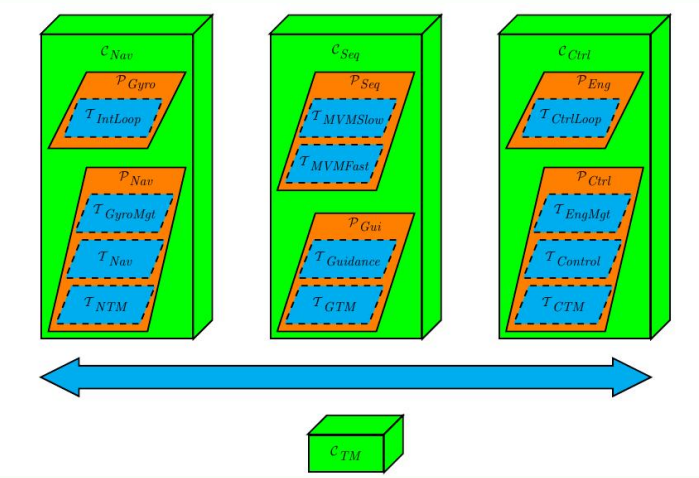
- architecture répartie sur 3 processeurs ($\mathcal{C}_{\text{Nav}}, \mathcal{C}_{\text{Seq}}, \mathcal{C}_{\text{Ctrl}}$) : traitement de l'information venant des capteurs, analyse des données, envoi des données vers les actuateurs



- software sur chaque processeur organisé en partitions
  - chaque partition contient plusieurs tâches
  - chaque tâche est périodique, caractérisée par (O, C, T) correspondant à offset, temps d'exécution, période
  - les tâches d'une partition se préemptent (ordonnancement RM)
  - les tâches de $\neq$ partitions se préemptent (switch de partition)
- plusieurs sources de non-determinisme pour ordonnancement
  - switch de partition ; entrelacement des tâches exécutées par $\neq$ processeurs

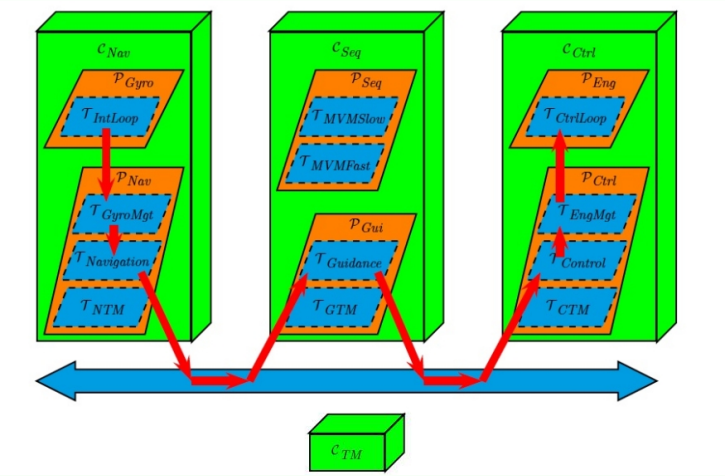
Enoncé du problème

- Étant donné une architecture processeur



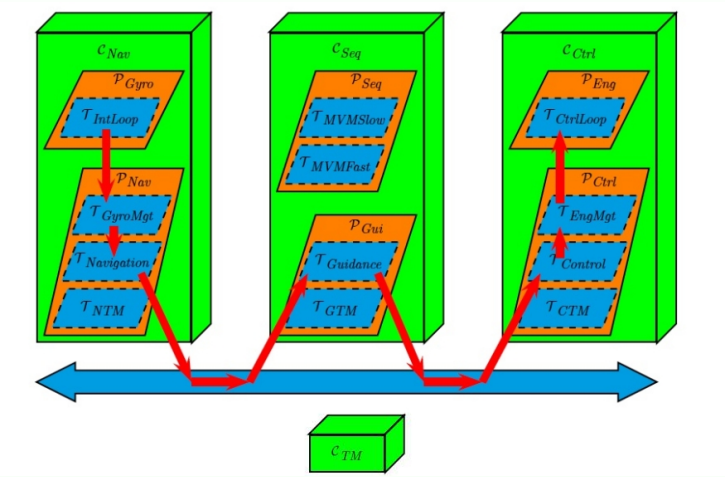
Enoncé du problème

- Étant donné une architecture processeur et un ensemble de travaux à effectuer



Enoncé du problème

- Étant donné une architecture processeur et un ensemble de travaux à effectuer



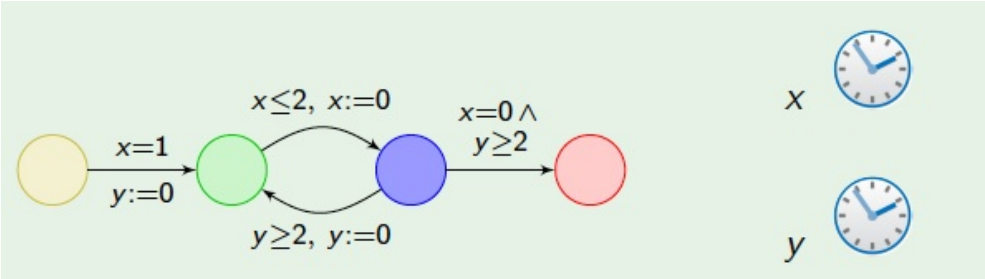
- Trouver un ordonnanceur qui assure que toutes les travaux seront effectuées avant une certaine date limite

Robustesse et ordonnançabilité

- source de complexité due au grand nombre de paramètres : priorités et dates limites des tâches, précédence
- un petit changement de la valeur d'un paramètre peut causer un changement complet de comportement
  $\rightsquigarrow$ nécessité de garantir la robustesse d'une solution d'ordonnancement
- plus généralement, un problème afférent important est de caractériser l'espace des paramètres qui préserve l'ordonnançabilité du système (toutes les tâches finissent avant leur date limite)

Approche LSV : raisonnement symbolique/ensembliste

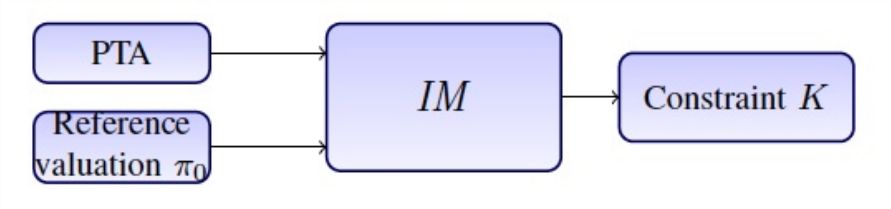
- modélisation des tâches et de l'ordonnanceur sous forme d'automates temporisés



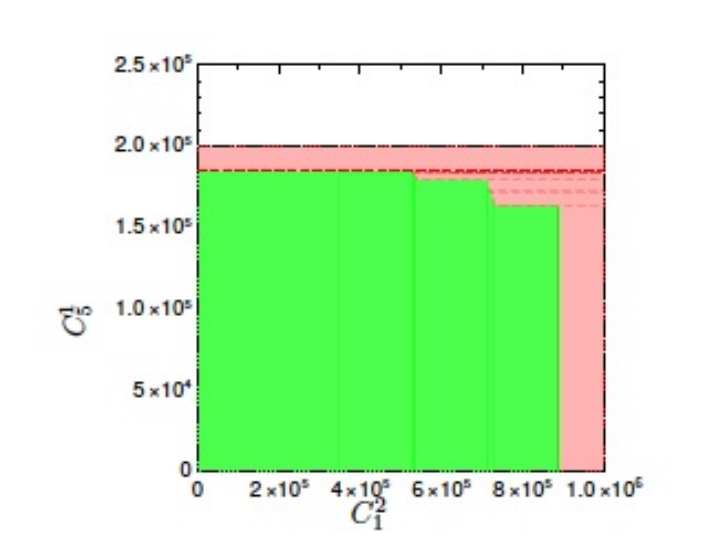
- réduction du problème de d'ordonnançabilité à un problème d'accessibilité
- traitement du problème d'accessibilité grâce à des techniques de model checking : représentation symbolique des données sous forme compacte avec des structures de données et des techniques d'exploration efficaces
- possibilité d'inclure des paramètres afin de générer des contraintes de robustesse

Approche du LSV : robustesse et ordonnançabilité

- Génération d'un ordonnancement pour une valeur donnée π_0 des paramètres, par exploration de l'espace des états accessibles
- génération d'une contrainte de robustesse autour de π_0 pour laquelle la correstion de l'ordonnancement est garantie

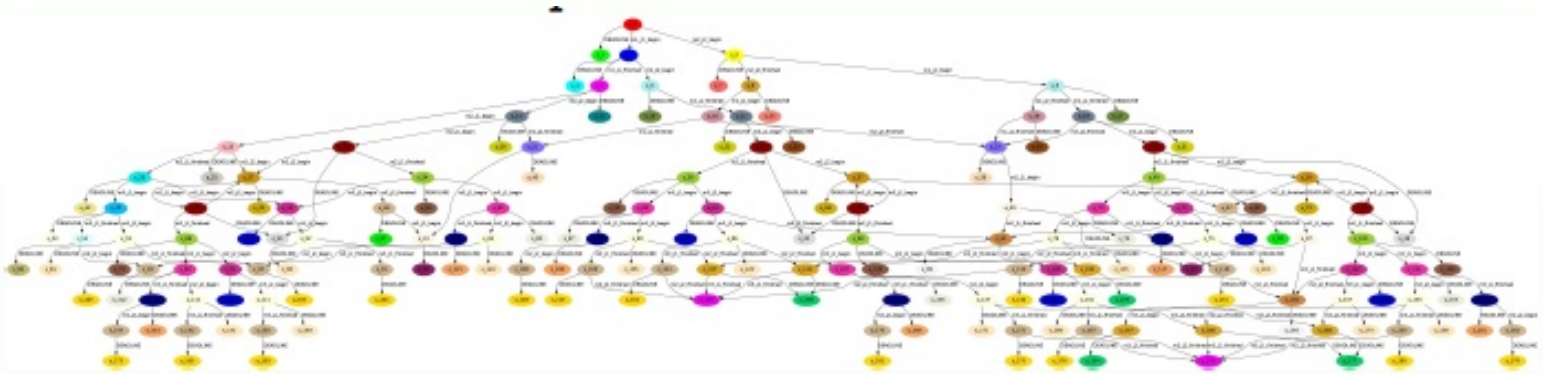


- par itération (pavage), génération du sous-espace ordonnançable

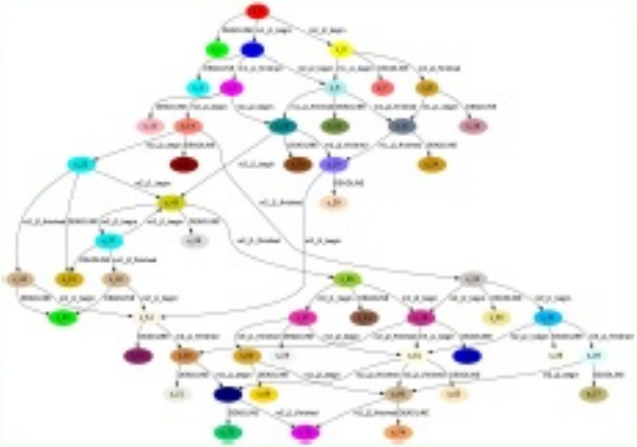


Effet de bord : amélioration de l'outil IMITATOR

- Exploration en force brute

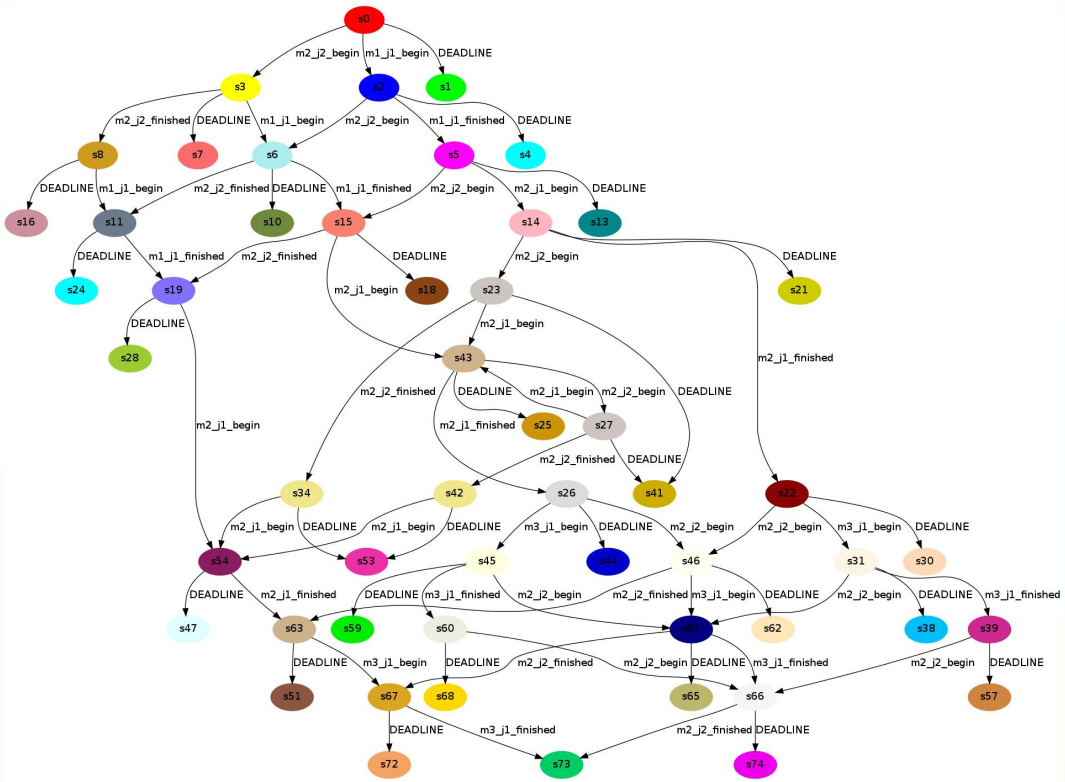


- Réduction de l'espace des états par fusion



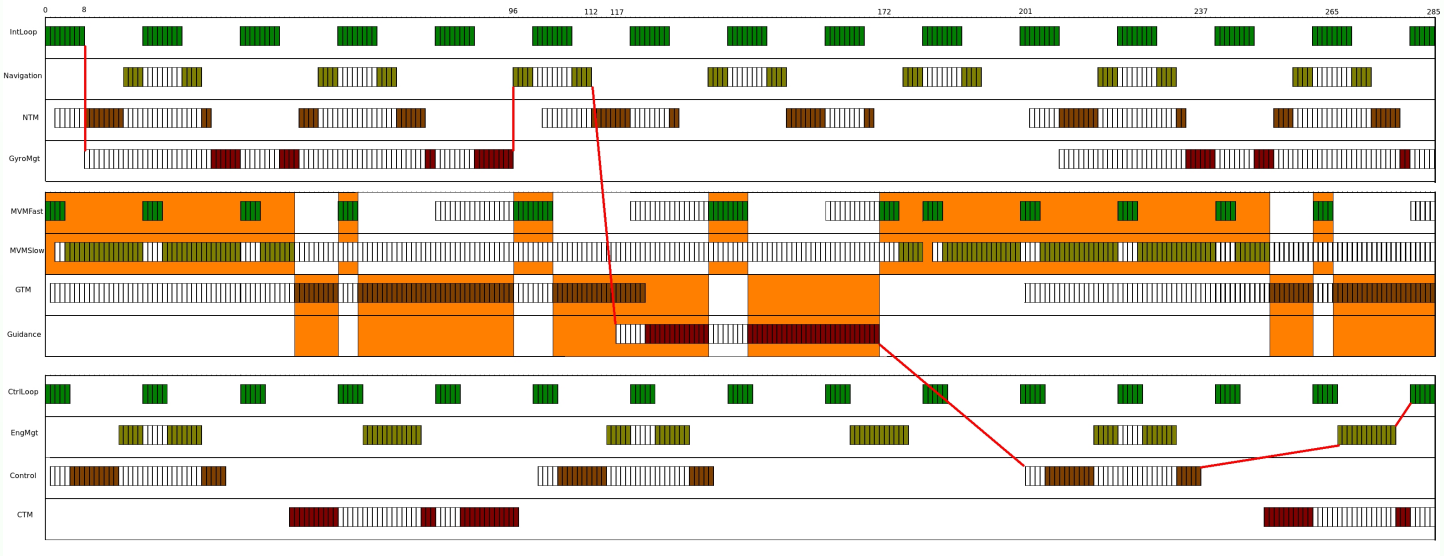
Traitement de l'étude de cas ASTRIUM

- Exploration de tous les ordonnanceurs possibles grâce á l'outil IMITATOR 2.5



Traitement de l'étude de cas Astrium (2)

- Visualisation de l'ordonnancement



Traitement de l'étude de cas ASTRIUM (3)

- Paramétrisation de toutes les grandeurs d'intérêt et génération de la contrainte de robustesse (via IMITATOR 2.5) :

$$
\begin{array}{ll}
 & 4 \geq C_{MVMFast} > 1 \\
\wedge & \wedge\ 8 \cdot C_{MVMFast} + C_{MVMSlow} > 71 \\
\wedge & \wedge\ C_{GTM} > 57 \\
\wedge & \wedge\ 100 \geq C_{Guidance} + C_{GTM} > 89 \\
\wedge & \wedge\ 120 > O_{Guidance} \geq 55 + C_{GTM} \\
\wedge & \wedge\ O_{Guidance} > C_{MVMFast} + C_{MVMSlow} \\
\wedge & \wedge\ C_{MVMFast} > O_{MVMSlow} > O_{GTM} > 0 \\
\wedge & \wedge\ O_{MVMFast} = 0
\end{array}
$$

- Tant que les paramètres satisfont cette contrainte, l'ordonnanceur trouvé auparavant satisfait les spécifications requises par Astrium.

Conclusion et travaux futurs

- Conclusion
  - découverte d'un ordonnancement original sur une maquette
  - garantie de robustesse de la solution
  - Amélioration induite sur IMITATOR par l'étude de cas
  - Réflexion sur la notion de ordonnançabilité
  - 2 publications dans des conférences internationales
    - *Robustness Analysis for Scheduling Problems using the Inverse Method*, in TIME 2012 (Temporal Representation and Reasoning)
    - *Parametric Schedulability Analysis of Fixed Priority Real-Time Distributed System*, in FTSCS 2013 (Formal Techniques for Safety-Critical Systems)
- Travaux futurs
  - Améliorations supplémentaires de l'outil IMITATOR pour passer à l'échelle et traiter une modélisation plus fine du l'architecture

Merci

Questions ?