

Analyses quantitatives de programmes

Séminaire LIG le 20 mars 2008

Laure Gonnord

<http://laure.gonnord.org/pro/>

Verimag/CITI/INSA
Grenoble/Villeurbanne, France



- 1 Contexte
- 2 Accélération Abstraite en Analyse des Relations Linéaires
- 3 Expression et Contractualisation de propriétés de Ressources

Systèmes embarqués

Plusieurs types de besoins :

- Programmes critiques : **vérification** du logiciel avant déploiement, propriétés de **sûreté**.
 - Programmes « multimédia » : développement rapide, garanties de **Qualité de Service** (QoS) à l'exécution.
- ▶ ou une combinaison !

Plan

- 1 Vérification de propriétés numériques de sûreté de programmes synchrones par Analyse des Relations Linéaires, amélioration par accélération.
 - ▶ **Thèse** de l'UJF, Grenoble, laboratoire Vérimag soutenue en octobre 2007. Sous la direction de N. Halbwachs.
- 2 Expression et garanties à l'exécution de propriétés de Qualité de Service dans des systèmes à composants.
 - ▶ **Postdoc** ANR depuis septembre 2007 - ARA Sécurité, systèmes embarqués et intelligence ambiante, Projet REVE. En collaboration avec J.-P. Babau.

- 1 Contexte
- 2 **Accélération Abstraite en Analyse des Relations Linéaires**
 - Analyse des Relations Linéaires
 - Problèmes de précision
 - Contributions
- 3 Expression et Contractualisation de propriétés de Ressources

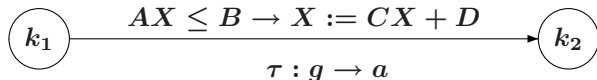
Vérification de propriétés de sûreté

Les propriétés :

- Propriétés de **sûreté**.
 - Propriétés numériques : **inéquations linéaires** $2y \leq 13$.
- ▶ Diverses méthodes : preuve, test, model-checking, **interprétation abstraite**.

Modèle - Notations

Vérification de propriétés **numériques** sur des GFC avec conditions et actions **affines** :

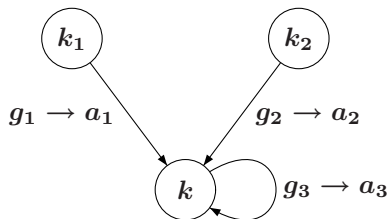


ou encore « automates interprétés », automates « à compteurs ».

- A, C matrices, B, D vecteurs.
- Sémantique « naturelle ».
- On veut des invariants pour **chaque point de contrôle**.

Formalisation du problème

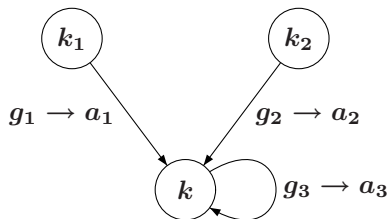
\mathcal{A}_k = ensemble des **valuations** au point k :



$$\mathcal{A}_k = a_1(\mathcal{A}_{k_1} \cap g_1) \cup a_2(\mathcal{A}_{k_2} \cap g_2) \cup a_3(\mathcal{A}_k \cap g_3)$$

Formalisation du problème

\mathcal{A}_k = ensemble des **valuations** au point k :



$$\mathcal{A}_k = a_1(\mathcal{A}_{k_1} \cap g_1) \cup a_2(\mathcal{A}_{k_2} \cap g_2) \cup a_3(\mathcal{A}_k \cap g_3)$$

- ▶ Système d'équations $X = F(X)$, **point fixe**.
- ▶ $X_0, F(X_0), F(F(X_0)), \dots$
 - Représentation des valuations, calcul.
 - Convergence de la résolution.

Résolution du système de point-fixe

Rappel des problèmes :

- Représentation, calcul
- Convergence de la résolution

Résolution du système de point-fixe

Rappel des problèmes :

- Représentation, calcul **polyèdres convexes**
- Convergence de la résolution **opérateur d'élargissement**

Résolution du système de point-fixe

Rappel des problèmes :

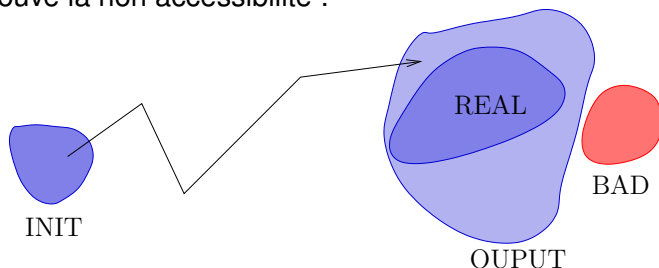
- Représentation, calcul **polyèdres convexes**
- Convergence de la résolution **opérateur d'élargissement**
- ▶ Résolution **approchée** mais **qui converge**

Résolution du système de point-fixe

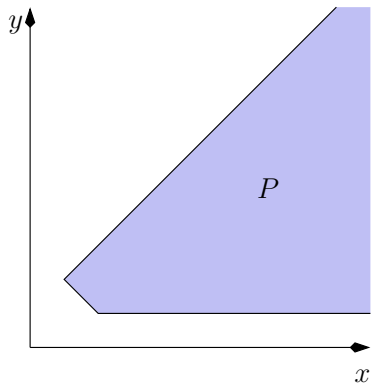
Rappel des problèmes :

- Représentation, calcul **polyèdres convexes**
- Convergence de la résolution **opérateur d'élargissement**
- ▶ Résolution **approchée** mais **qui converge**

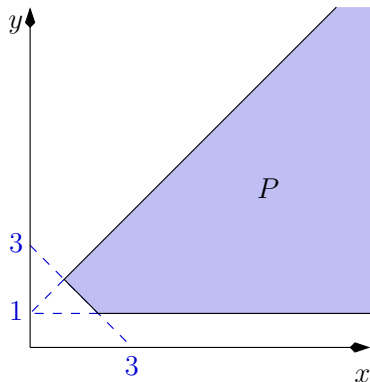
On prouve la non accessibilité :



Le treillis des polyèdres - Double Représentation

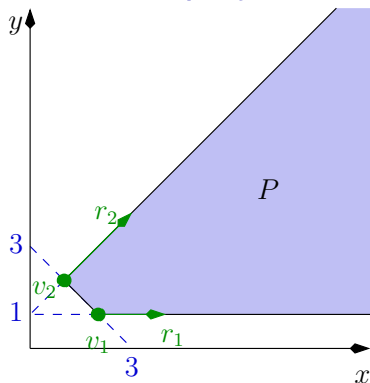


Le treillis des polyèdres - Double Représentation



$$\begin{aligned} P &= \{(x, y) \mid \\ &\quad 1 \leq y \leq x + 1 \wedge x + y \geq 3\} \\ &= \text{cons}\{AX \leq b\} \end{aligned}$$

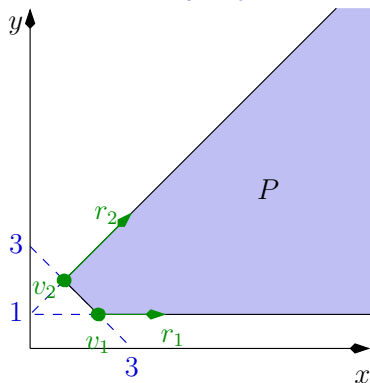
Le treillis des polyèdres - Double Représentation



$$\begin{aligned}
 P &= \{(x, y) \mid \\
 &\quad 1 \leq y \leq x + 1 \wedge x + y \geq 3\} \\
 &= \text{cons}\{AX \leq b\}
 \end{aligned}$$

$$\begin{aligned}
 P &= \{\lambda v_1 + (1 - \lambda)v_2 + \mu_1 r_1 + \mu_2 r_2 \mid \\
 &\quad \lambda \in [0, 1], \mu_1, \mu_2 \geq 0\} \\
 &= \text{gen}(V, R)\}
 \end{aligned}$$

Le treillis des polyèdres - Double Représentation



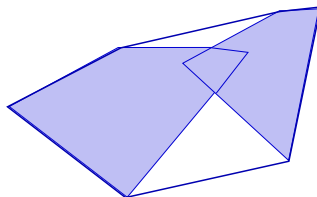
$$\begin{aligned}
 P &= \{(x, y) \mid \\
 &\quad 1 \leq y \leq x + 1 \wedge x + y \geq 3\} \\
 &= \text{cons}\{AX \leq b\}
 \end{aligned}$$

$$\begin{aligned}
 P &= \{\lambda v_1 + (1 - \lambda)v_2 + \mu_1 r_1 + \mu_2 r_2 \mid \\
 &\quad \lambda \in [0, 1], \mu_1, \mu_2 \geq 0\} \\
 &= \text{gen}(V, R)\}
 \end{aligned}$$

- Deux représentations **finies** et complémentaires (passage de l'une à l'autre).
- Algorithmique disponible.

Le treillis des polyèdres convexes (2)

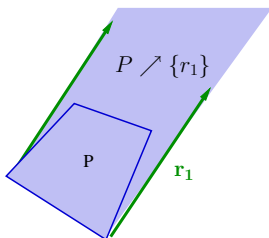
- Intersection, test du vide.
- Transformation affine : $a(P) = \{CX + D \mid X \in P\}$.
- Union convexe (perte de précision) :



Le treillis des polyèdres convexes (2)

- Intersection, test du vide.
- Transformation affine : $a(P) = \{CX + D \mid X \in P\}$.
- Union convexe (perte de précision) :
- Ajout de rayons

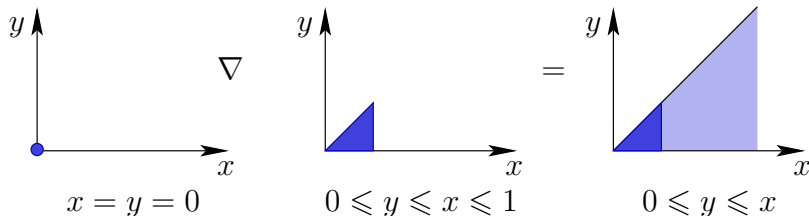
$$P \nearrow R = \left\{ X + \sum_{r_j \in R} \mu_j r_j \mid X \in P, \mu_j \in \mathbb{Q}^+ \right\}$$



Le treillis des polyèdres convexes (3)

Élargissement : $P_1 \nabla P_2$: extrapolation de la limite.

Le système de contrainte de $P \nabla Q$ est obtenu en enlevant du système de P les contraintes non satisfaites par Q :



Astuce (!) : $\{x = y = 0\} = \{0 \leq y \leq x \leq 0\}$

Les problèmes de l'Analyse des Relations Linéaires

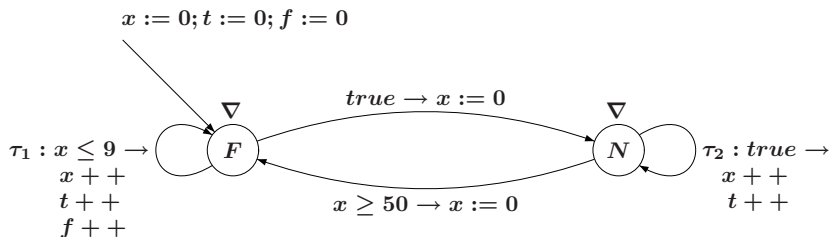
Sources de complexité :

- nombre de points de contrôle.
- nombre de variables numériques.

Sources d'approximation :

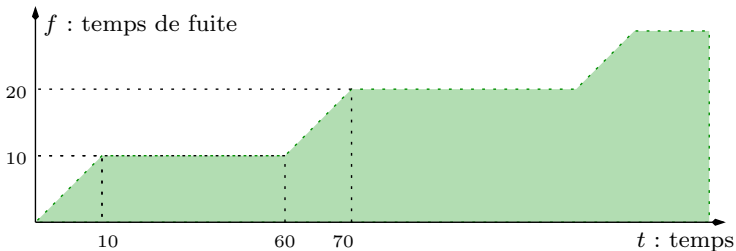
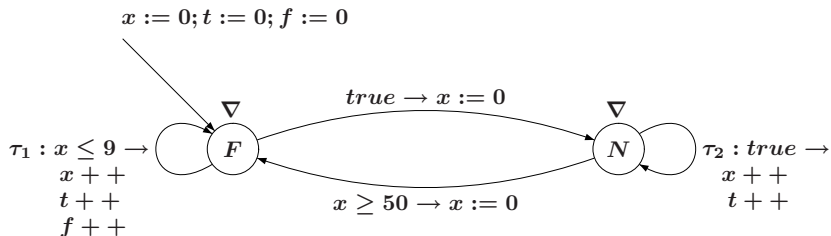
- Enveloppe convexe.
- **Élargissement.**

L'exemple de la chaudière - 1

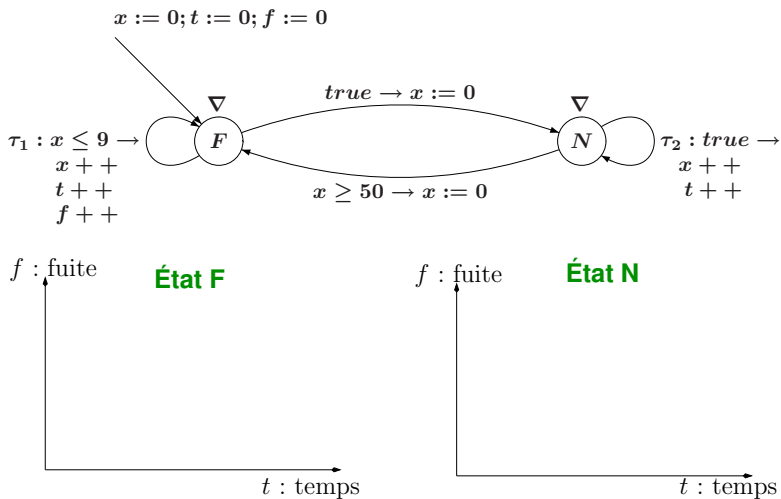


- f le temps de fuite global.
- t le temps global.
- x variable locale.

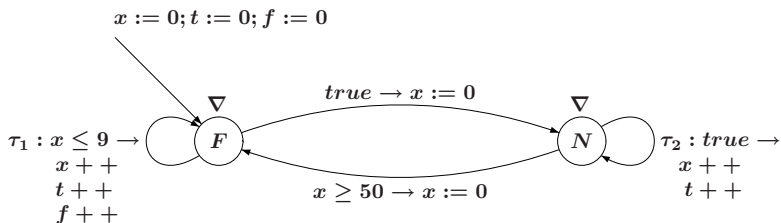
La chaudière 2 - Comportement réel



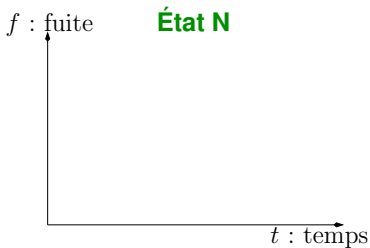
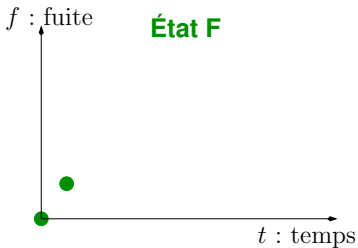
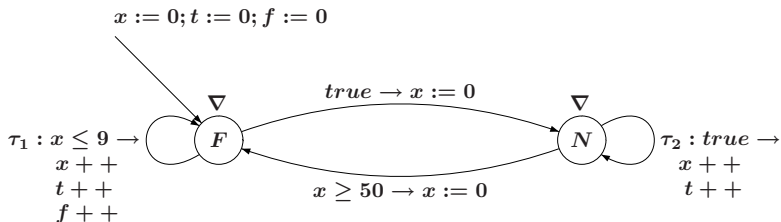
La chaudière 3 - Analyse des Relations Linéaires



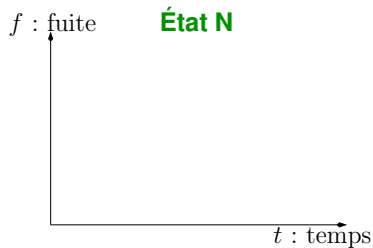
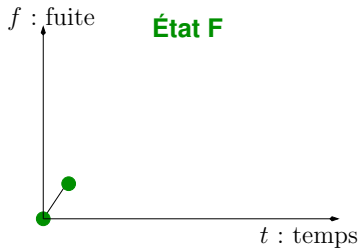
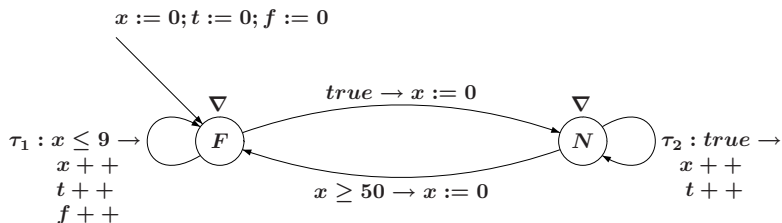
La chaudière 3 - Analyse des Relations Linéaires

 $f : fuite$ **État F** $f : fuite$ **État N** $t : temps$ $t : temps$

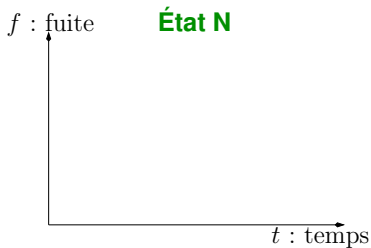
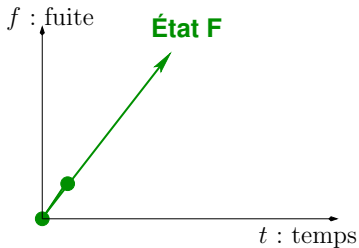
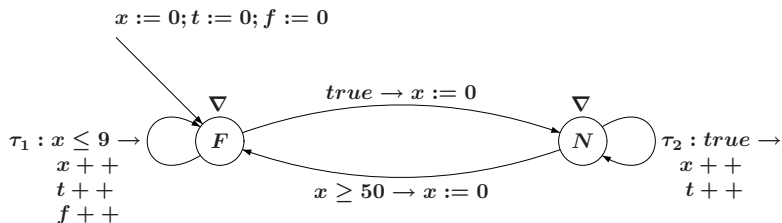
La chaudière 3 - Analyse des Relations Linéaires



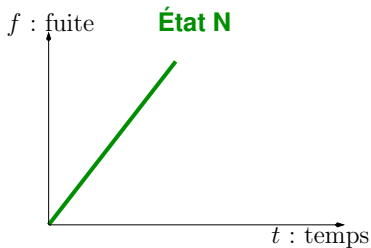
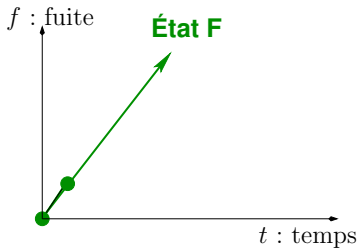
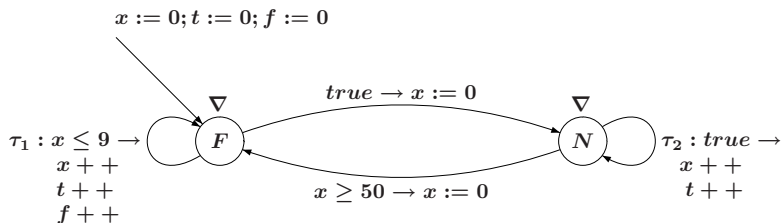
La chaudière 3 - Analyse des Relations Linéaires



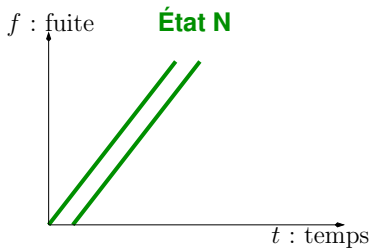
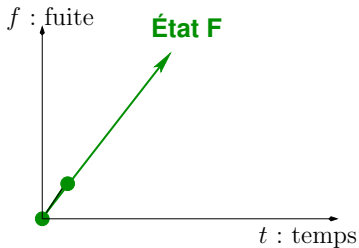
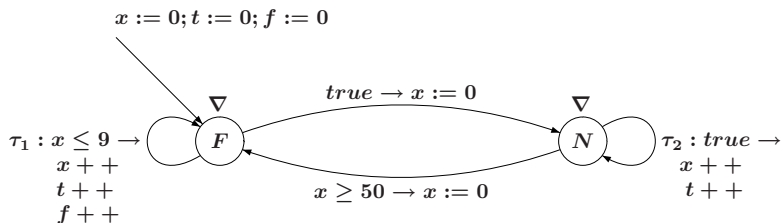
La chaudière 3 - Analyse des Relations Linéaires



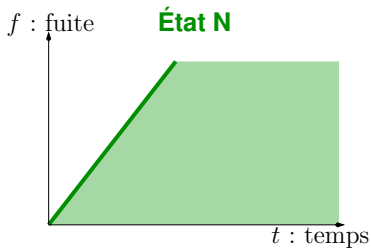
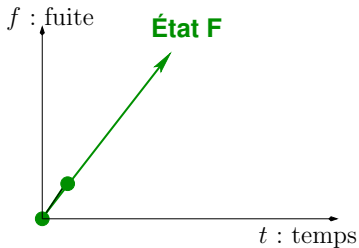
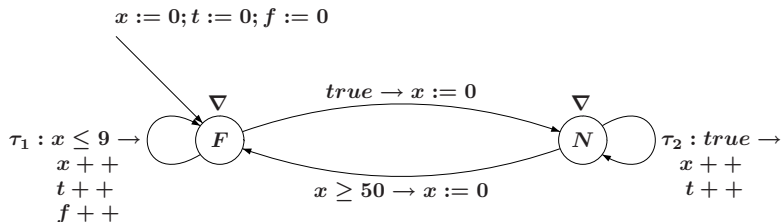
La chaudière 3 - Analyse des Relations Linéaires



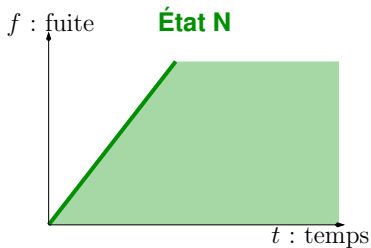
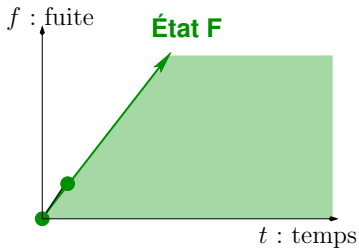
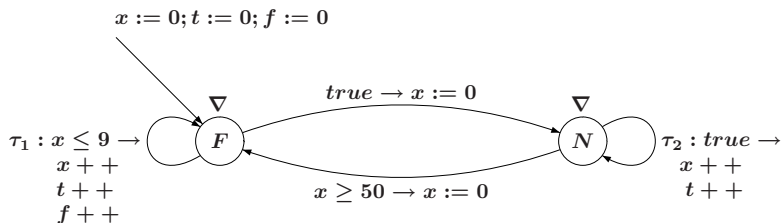
La chaudière 3 - Analyse des Relations Linéaires



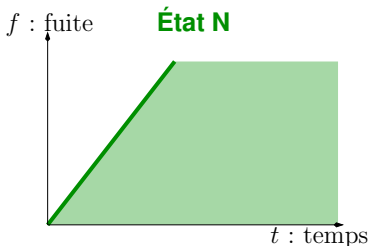
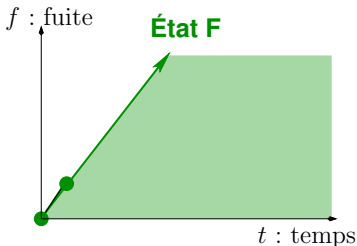
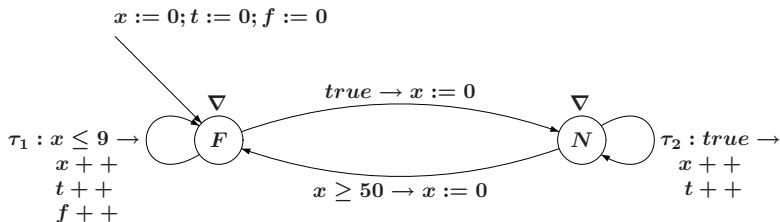
La chaudière 3 - Analyse des Relations Linéaires



La chaudière 3 - Analyse des Relations Linéaires

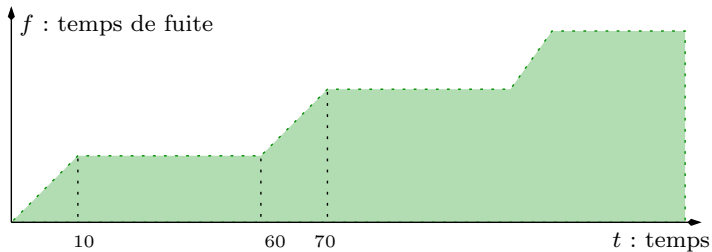
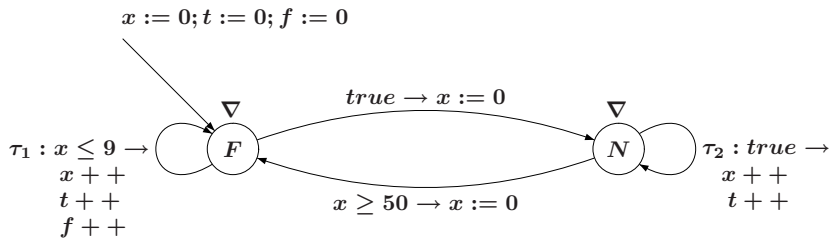


La chaudière 3 - Analyse des Relations Linéaires

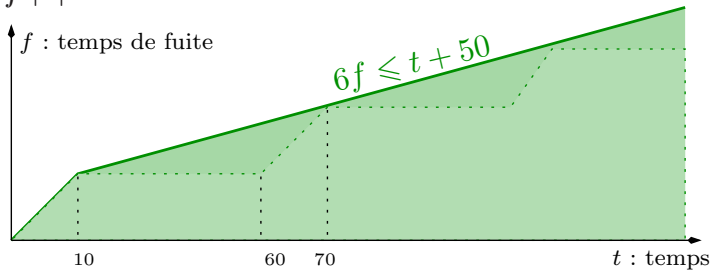
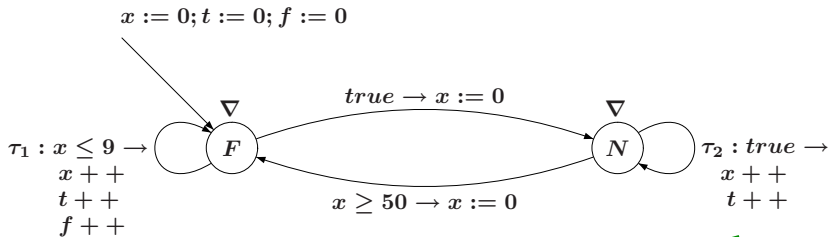


- Convergence, approximation supérieure, mais manque de précision.

La chaudière - Invariant voulu



La chaudière - Invariant voulu



Amélioration de la précision

Des méthodes existent mais sont **adhoc** (en général).

- ▶ Regardons les méthodes **exactes (accélération)** :
 - On calcule l'effet exact des boucles sur des ensembles d'**entiers**.
 - Codage sous forme d'automates représentant des formules de Presburger, par ex : $\exists k, x = y + 2k$.

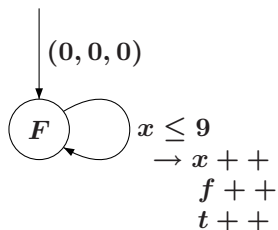
Amélioration de la précision

Des méthodes existent mais sont **adhoc** (en général).

- ▶ Regardons les méthodes **exactes (accélération)** :
 - On calcule l'effet exact des boucles sur des ensembles d'**entiers**.
 - Codage sous forme d'automates représentant des formules de Presburger, par ex : $\exists k, x = y + 2k$.
- ▶ **Inconvénients** : classes restreintes de programmes, haute complexité.

Exemple de la chaudière (3)

La boucle est « accélérable » :



► Effet **exact** : $\exists i \in \mathbb{N}, x = f = t = i, 0 \leq i \leq 10$

Contribution : théorie, algorithmique

Définition de la notion d'**accélération abstraite** qui :

- Permet d'obtenir des approximations supérieures **précises** à faible coût.
- Se combine bien avec l'élargissement.

Contribution : théorie, algorithmique

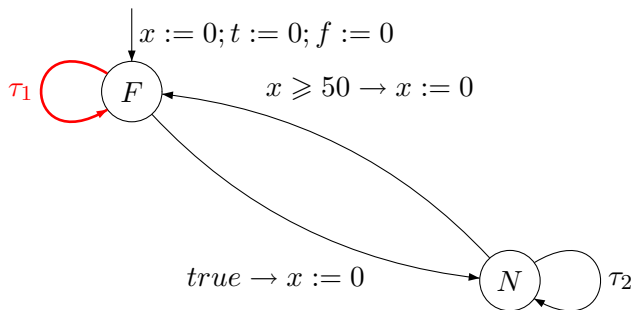
Définition de la notion d'**accélération abstraite** qui :

- Permet d'obtenir des approximations supérieures **précises** à faible coût.
- Se combine bien avec l'élargissement.

▶ cf SAS 2006

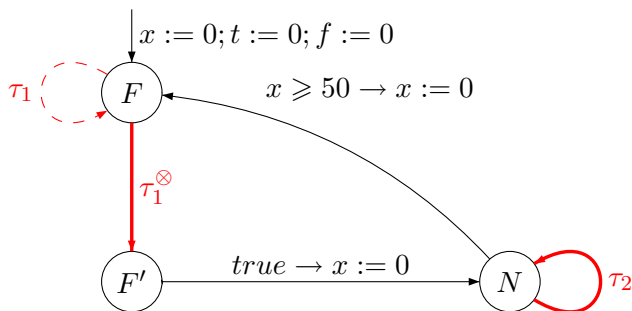
Accélération et partitionnement

On remplace les boucles ($\tau_i : g_i \rightarrow a_i$)



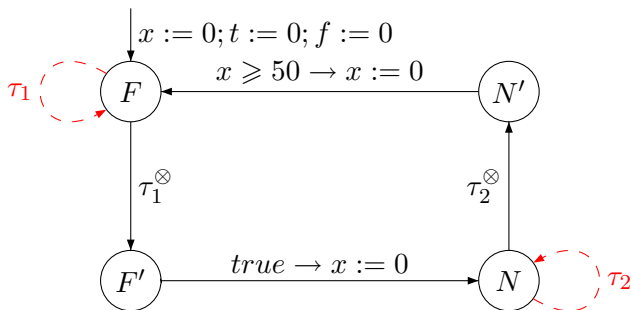
Accélération et partitionnement

On remplace les boucles ($\tau_i : g_i \rightarrow a_i$)



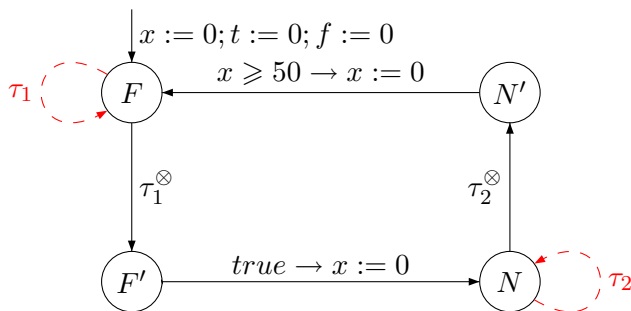
Accélération et partitionnement

On remplace les boucles ($\tau_i : g_i \rightarrow a_i$)



Accélération et partitionnement

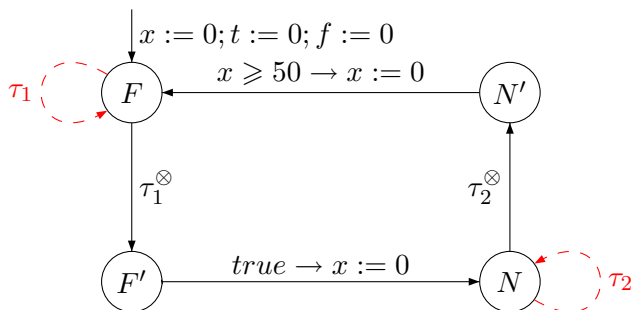
On remplace les boucles ($\tau_i : g_i \rightarrow a_i$)



► τ_i^\otimes résume l'effet d'une application de τ_i un **nombre quelconque de fois**

Accélération et partitionnement

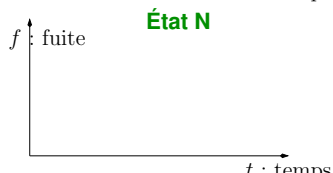
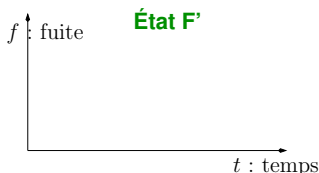
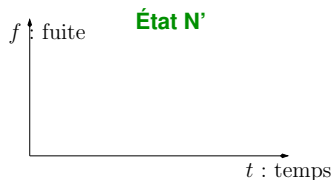
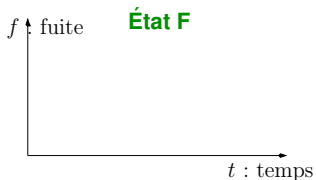
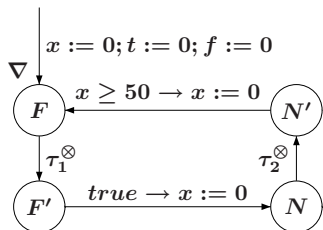
On remplace les boucles ($\tau_i : g_i \rightarrow a_i$)



► τ_i^\otimes résume l'effet d'une application de τ_i un **nombre quelconque de fois**

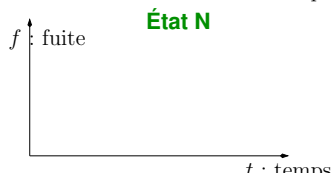
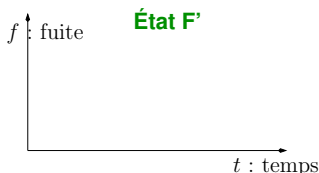
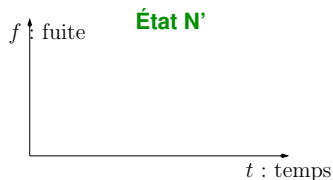
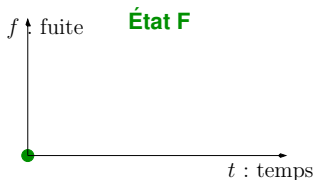
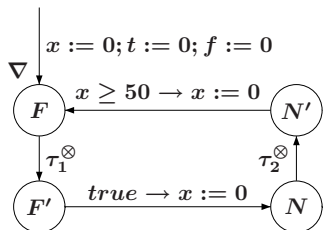
► Boucle englobante : **accélérée** ou **élargie**.

Le retour de la chaudière



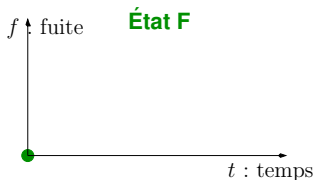
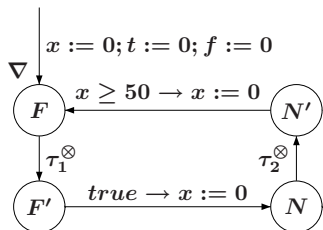
- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

Le retour de la chaudière

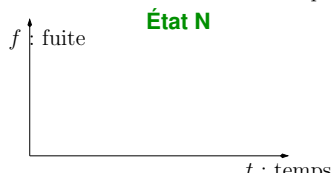
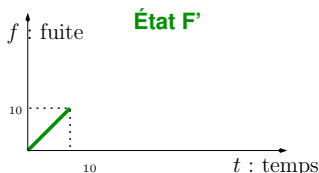
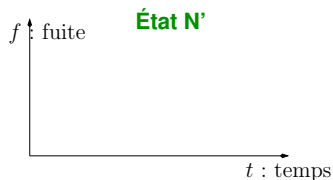


- τ_1^{\otimes} = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^{\otimes} = “ajoute le rayon (1, 0, 1)”

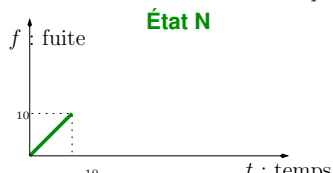
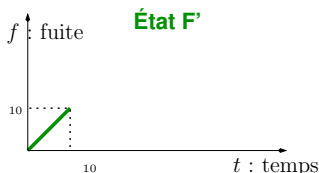
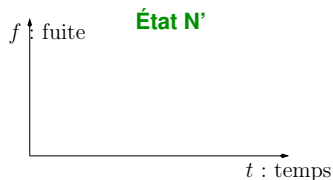
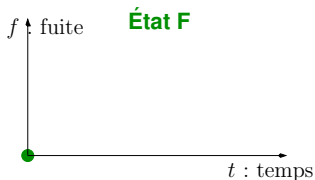
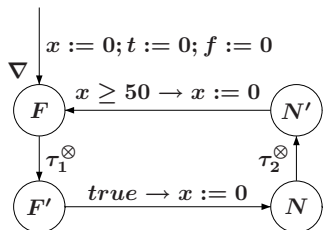
Le retour de la chaudière



- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

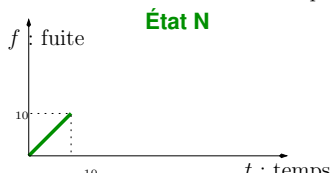
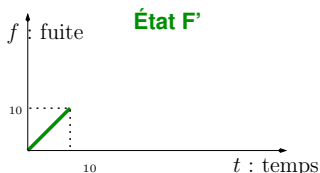
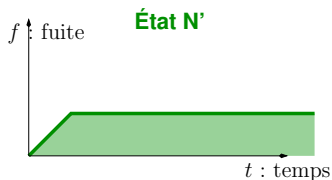
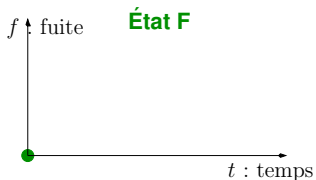
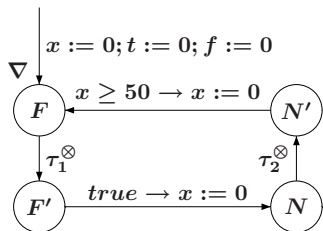


Le retour de la chaudière



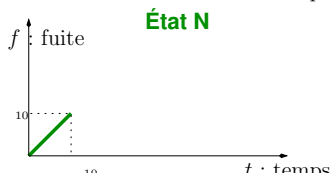
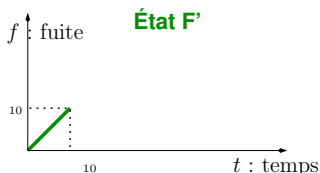
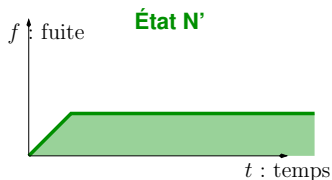
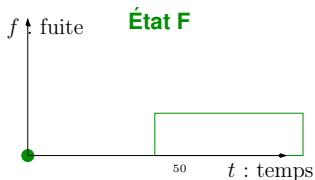
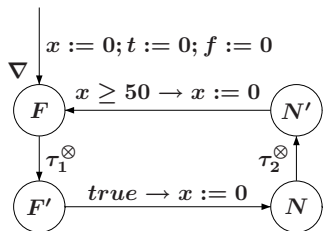
- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

Le retour de la chaudière



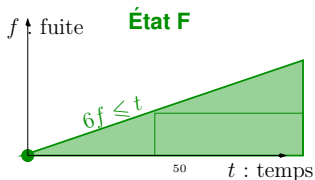
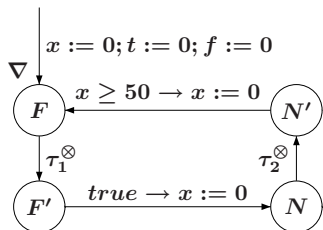
- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

Le retour de la chaudière

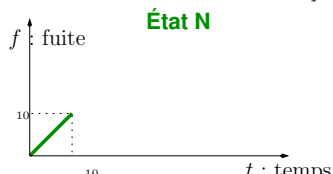
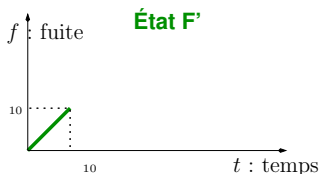
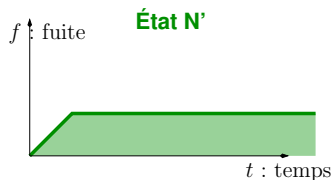


- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

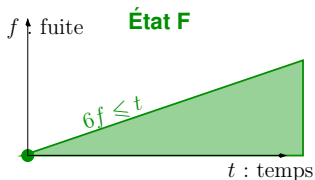
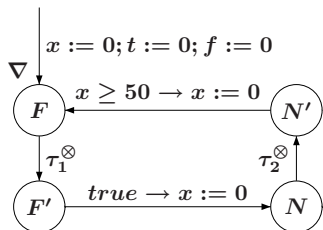
Le retour de la chaudière



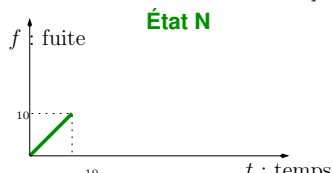
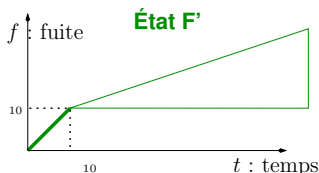
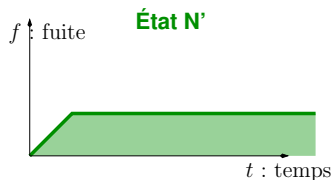
- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”



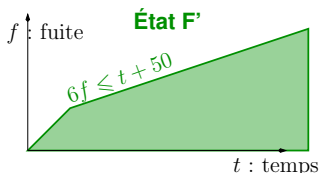
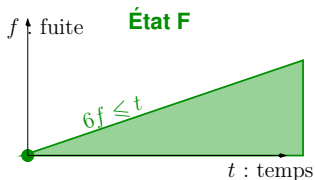
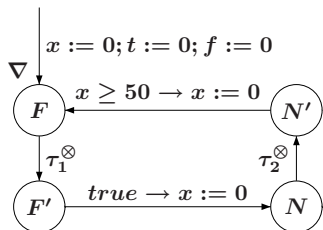
Le retour de la chaudière



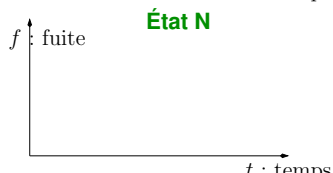
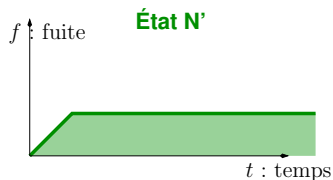
- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”



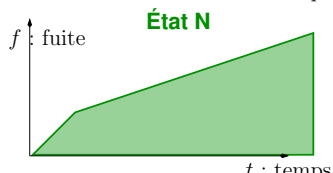
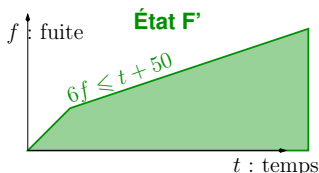
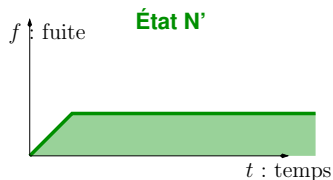
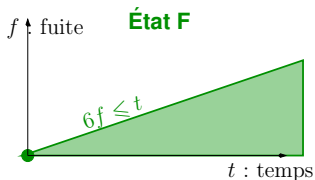
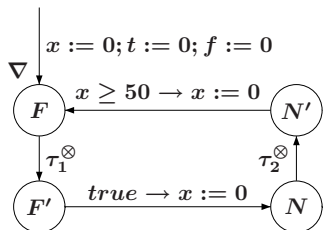
Le retour de la chaudière



- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

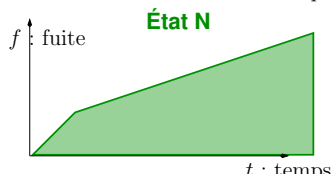
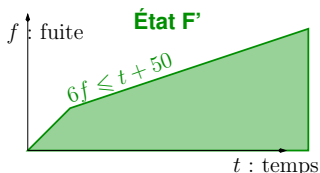
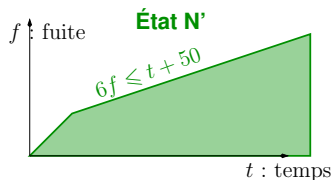
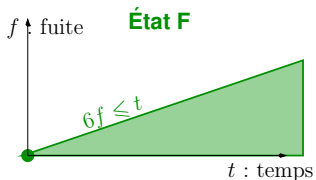
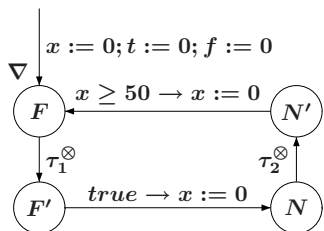


Le retour de la chaudière



- $\tau_1^\otimes =$ “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- $\tau_2^\otimes =$ “ajoute le rayon (1, 0, 1)”

Le retour de la chaudière



- τ_1^\otimes = “ajoute le rayon (1, 1, 1) tant que $x \leq 10$ ”
- τ_2^\otimes = “ajoute le rayon (1, 0, 1)”

Prototype développé : Aspic

ASPIC : Accelerated Symbolic Polyhedral Invariant
Computation

- Un langage textuel d'automates (Fast) avec ou sans but de preuve (formule)
- Calcul classique + accélérations.
- Sorties : invariants + diagnostic.

Prototype développé : Aspic

ASPIC : Accelerated Symbolic Polyhedral Invariant Computation

- Un langage textuel d'automates (Fast) avec ou sans but de preuve (formule)
- Calcul classique + accélérations.
- Sorties : invariants + diagnostic.

▶ <http://laure.gonnord.org/pro/aspic/aspic.html>

Résultats expérimentaux

Quelques applications

- Accessibilité dans des automates à compteurs (sémantique de SystemC), une centaine de points de contrôle, J. Cornet.
- Propriétés numériques d'automates modélisant une consommation d'énergie de (réseaux de) capteurs, L. Samper et F. Maraninchi.
- Vérification de programmes manipulant des listes, R. Iosif et S. Perarnau.

Résultats expérimentaux

Quelques applications

- Accessibilité dans des automates à compteurs (sémantique de SystemC), une centaine de points de contrôle, J. Cornet.
 - Propriétés numériques d'automates modélisant une consommation d'énergie de (réseaux de) capteurs, L. Samper et F. Maraninchi.
 - Vérification de programmes manipulant des listes, R. Iosif et S. Perarnau.
- ▶ Cf ma thèse et le futur papier de journal.

- 1 Contexte
- 2 Accélération Abstraite en Analyse des Relations Linéaires
- 3 Expression et Contractualisation de propriétés de Ressources
 - Contexte - hypothèses
 - Qinna
 - Formalisation de Qinna++ ; mise en œuvre
 - En cours

Nos composants

Caractéristiques

- Un composant fournit des **services** (fonctions, paramètres). Un service peut avoir des **requis** (services d'un autre composant).
- On distingue **type** de composant (mémoire,cpu,screen) et **instance** de composant.
- Notion de **liaison** entre les composants (services requis/fournis). Pas de mémoire partagée.

Composants, concepts pour l'embarqué

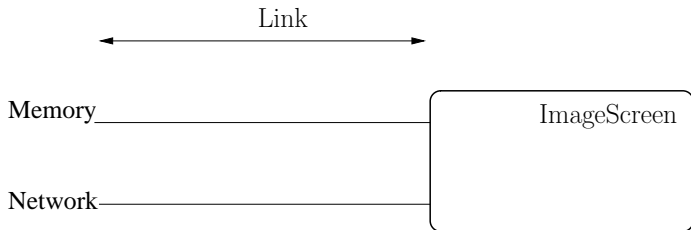
Ressources bornées :

- Contraintes numériques, ressources monitorables.
- Ressources : **composants**.
- Donner des « autorisations d'accès » aux services selon la consommation.

Composants, concepts pour la QoS (qualité de service)

Ressources variables :

- différentes implémentations des services selon la « qualité de service » (**niveau d'implémentation**)
- adaptation de ces implémentations par **contrat de liaison** (contrat = lien entre des parties).



Objectif

On veut **contractualiser des propriétés** des services :

- Exprimer.
- Garantir.
- Négocier.

Les propriétés de qualité de service (QoS) portent sur les composants et les services, parlent de ressources (pas de délai pour le moment).

► Une **architecture** pour la gestion de la qualité de service (Qinna).

Qinna en Bref - Architecture Logicielle

Une architecture logicielle dédiée à la gestion des problèmes de **Qualité de Service**. Nouveaux composants :

- `QoSComponent` : service(s) à qualité variable.
 - `QoSBroker` : initialisation et réservation des composants
 - `QoSManager` : négociation, contractualisation
 - `QoSDomain` : ensemble des demandes de services.
- ▶ cf thèse de J.-C. Tournier (CITI/France Télécom).

Qinna première version

- ☺ Séparation des préoccupations (composants).
- ☺ Lien entre les différentes QoS par tables.
- ☺ Procédé de négociation générique.
- ☹ Pas de distinction type (classe)/instance/service.
- ☹ Pas de distinction propriété de comportement/de liaison.
- ☹ Pas d'implémentation générique.
- ☹ Pas de procédé d'évaluation.

► Qinna++

Deux classes de propriétés

Nous avons identifié deux classes de propriétés (de QoS) :

- des propriétés sur **les services et les composants** (QoS Behavioural Constraints) : « la mémoire totale pour l'application est 4Mo », « le service `imagedisplay` est monutilisateur », « ce composant ne peut être alloué que 3 fois », ...
- des propriétés sur les **liaisons** (QoS Linking Constraints) : « pour que le service `imagedisplay` soit rendu à qualité maximale, il faut que le réseau soit rapide et que le taux de compression soit faible »...

Formules sur les **paramètres** d'appels des services, sur la suite des **occurrences** des services,

Contraintes de comportement de ressources (QBC)

- CTC : **contrainte de type** (nb total d'instances, ressource totale pour tous les services du composant, ...) :

$$CTC(C) \stackrel{def}{=} \bigwedge_i CTC_{serv}(s_i) \wedge CTC_{compo}(C)$$

- CIC : **contrainte d'instance** (idem sur l'instance) :

$$CIC(C^j) \stackrel{def}{=} \bigwedge_i CIC_{serv}(s_i^j) \wedge CIC_{instance}(C^j)$$

(C : composant qui fournit le service s , C^j et s^j instances.)

Mise en œuvre des QBC dans Qinna++

Pour le (type) QoSComponent S , on crée un QoSBroker : les QBC seront garanties .

- Le Broker fait l'allocation d'un composant uniquement si les CTC sont vérifiées (procédure de décision dans le Broker). Il initialise les contraintes d'instance (CIC).
- Le Broker sait décider si une instance de composant est en mesure de fournir un service à une quantité de service donnée.
- Après l'appel à un service, on lance `upgradeCIC` dans l'instance, et `upgradeCTC` dans le Broker.

Ressources bornées ► décision à **mémoire bornée**.

Mise en œuvre des QLC dans Qinna++

Utilisation du composant QoSManager (pour S) ·

- Expression : une **table de liaison** lie les niveaux d'implémentation des services et de leurs requis.
- Lors de la réservation d'un service à une certaine qualité objectif, le Manager utilise les infos de la table pour établir (éventuellement) un **contrat**.

Réservation d'un service à un certain niveau de QoS/d'implémentation

Mise en place du **contrat de liaison** pour le service `Screen.displayimage()` (l'image existe) d'une l'instance à une QoS objectif .

- Le Broker est déjà initialisé (CTC), et une instance créée (CIC en partie). Si non CIC(s), échec.
 - On demande le service avec le niveau de QoS == TBon (ie `implLevel=1`). Le Manager dit qu'il faut *20ko* de mémoire : le service `Memory.malloc(20)` est demandé : si CIC et CTC ok, mémoire allouée.
 - Contrat == `<display,1,malloc,20>`
- ▶ Réalisé **automatiquement** par les Managers de Qinna++.

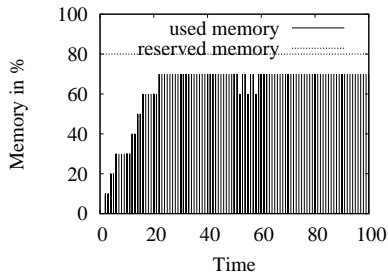
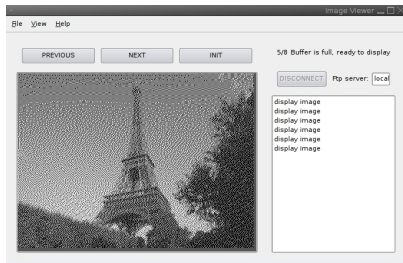
Dégradation en cas d'échec

Deux notions :

- Les **niveaux d'implémentation** d'un service donné (ordonnés) : un contrat peut comporter un intervalle de niveaux d'implémentation objectifs acceptable.
 - Les appels de services sont classés par **niveau d'importance**.
- ▶ Si la négociation échoue, on dégrade la QoS objectif (ie l'implémentation) d'un service de moindre importance.

En cours, et ensuite ... - 1

- Soumission à CORCS 2008 de l'étude sur les contraintes.
- Finalisation du cas d'étude, variations et monitoring.



En cours, et ensuite ... - 2

- Définition d'une méthodologie d'implémentation.
- Composants Qinna génériques et méta modèle pour Kermeta (IRISA).
- Bibliothèque de composants de ressources : mémoire, CPU, Thread, plus ou moins précis.
- Vers la vérification des propriétés numériques : composants IF et observateurs. (ou Aspic !)
- Génération automatique des Brokers (logique plus précise pour les CTC/CIC).
- Propriétés de délai.

Questions ?

Merci .