Test et Testabilité de Logiciels Synchrones

- Programmation synchrone, le langage Lustre
  Nicolas Halbwachs, Vérimag/CNRS, Grenoble
- Techniques de test de logiciels réactifs synchrones
  Farid Ouabdesselam, LSR/UJF, Grenoble
- Testabilité de systèmes flot-de-données
  Chantal Robach, LCIS/INPG/ESISAR, Valence
- Projets PSLC en cours, et conclusion

Programmation Synchrone
L'Approche "Flot de Données"

Le Language LUSTRE

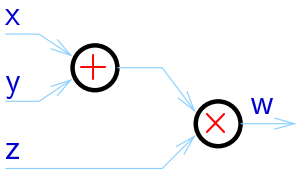
Nicolas Halbwachs
Vérimag/CNRS
Grenoble

Idée initiale [P. Caspi 84]:
Formalismes naturels des ingénieurs automaticiens

- Schémas-bloc
- Schémas analogiques
- Réseaux de portes, schémas à relais
- Equations différentielles, aux différences finies

Confirmation: existance de nombreux outils propriétaires basés sur ce type de formalisme

- Energie (Schneider-Electric, EDF) :
  "diagrammes fonctionnels"
- Avionique (Aérospatiale, Sextant, . . .)



$$w = (x + y) \times z$$

$$\forall t, w_t = (x_t + y_t) \times z_t$$
$$\forall n, w_n = (x_n + y_n) \times z_n$$

à chaque période faire
    lire(x); lire(y); lire(z);
    w := (x+y)*z;
    écrire(w);
refaire

Avantages

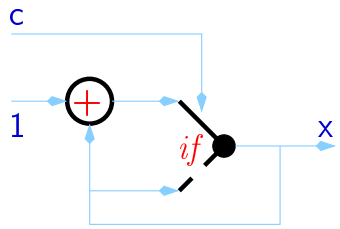
- culture de l'ingénieur
- "propreté" mathématique
- naturellement modulaire
- graphique
- naturellement parallèle

Besoins

- sémantique formelle précise
- structuration de programmes
- contrôles statiques (types, ...)
- génération automatique de code
- outils de validation

Boucles et retards

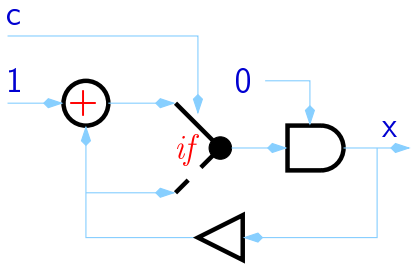
$x_n =$ if c_n then $x_{n-1} + 1$
else x_{n-1}



Problèmes:

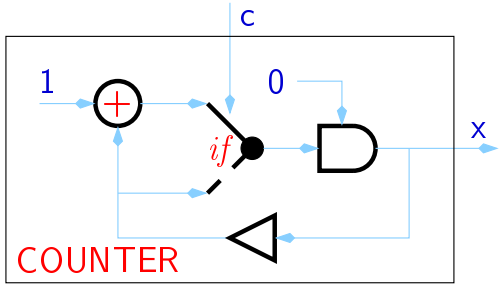
- référence à la valeur précédente: x_{n-1}
- initialisation: x_0

x = 0 -> if C then pre(x) + 1
else pre(x)

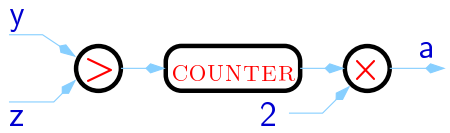


$$x_n = \begin{cases} 0 & \text{if } n = 0 \\ x_{n-1} + 1 & \text{if } n \neq 0 \text{ and } c_n = true \\ x_{n-1} & \text{if } n \neq 0 \text{ and } c_n = false \end{cases}$$

Structuration de programmes (Opérateurs définis par l'utilisateur)



a = 2*COUNTER(y> z)



Déclarations de Variables et de Noeuds
(Vérifications statiques de cohérence)

```
node COUNTER(c: bool) returns (x: int);
let
    x = 0 -> if C then pre(x) + 1 else pre(x);
tel
...
var a: int; y,z: real;
...
a = 2*COUNTER(y>z);
...
```

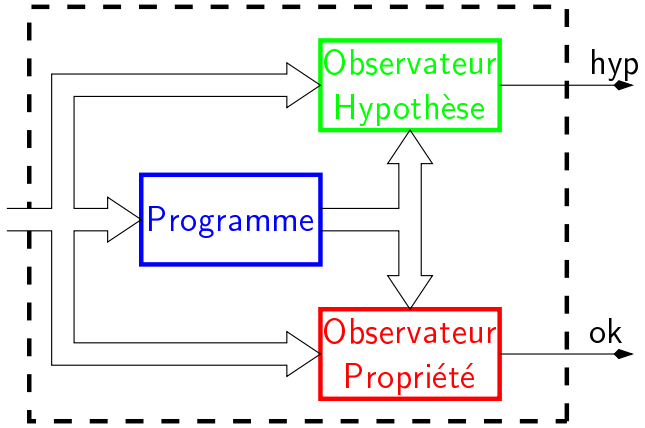

Vérification de programmes

Objectif:

- exprimer les propriétés critiques attendues
- exprimer les hypothèses sur l'environnement
- vérifier automatiquement (si possible) ces propriétés sous ces hypothèses

Expression d'une propriété (ou d'une hypothèse)
par un observateur synchrone

observateur = programme recevant les variables
significatives et calculant
une sortie booléenne "ok",
vraie tant que la propriété est satisfaite.



Montrer ok toujours vrai, ou hyp faux avant.

Vérification par examen exhaustif du comportement des variables booléennes (+ certains aspects numériques simples)

Méthode automatique mais approchée : possibilité de résultats pessimistes.

Un outil : LESAR

EXPÉRIMENTATIONS EN VRAIE GRANDEUR
(SCHNEIDER-ELECTRIC, AEROSPATIALE, ...)

Développements et Industrialisation