UNIVERSITE BISKRA – ALGERIE
FACULTE DES SCIENCES EXACTES ET DES SCIENCES DE LA NATURE ET DE LA VIE ,
DEPARTEMENT D'INFORMATIQUE
Master 2 GLSD

LOGIQUE DE REECRITURE:
MODELISATION DES SYSTEMES DYNAMIQUES

présenté par : Mohamed RAMDANI

Le contenu de ces *slides* est une traduction d'un support de cours du module INF3032 présenté à l'université d'Oslo par Rudolf Schlatte

SYSTEME DYNAMIQUE (1)

Système Dynamique : Système qui évolué

- Changement d'état.

Exemples :

- Système d'exploitation,
- Système d'échange de messages dans un réseau
- Programme de contrôle d'une usine d'energie,
- système de compte bancaires
- . . .

SYSTEME REACTIF

Système d'exploitation et les systèmes de contrôle du trafique airain sont des exemples de systèmes dynamiques et réactifs.

- Ils réagirent aux entrées issues de l'environnement,
- .. par changement d'état et/ou par produire des sorties vers l'environnement,
- Généralement, ne sont pas systèmes terminant,
- Souvent, ne sont pas déterministes,
- les paternes d'interaction et de changements d'état sont des concepts intéressant dans tels système,

SYSTEME STATIQUES

Les systèmes Statiques / types de données complexe sont définies par des équations,

- Mathématiques : logique équationnelle :
  - …|-- s(0) + s(0) = s(s(0))
  - …|-- s(s(0)) = s(0) + s(0)
  - …|-- length (2 5 7) = 3
  - …|-- 3 = length (2 5 7)

- Symétrie, tout les changements sont réversibles

SYSTEME DYNAMIQUE (2)

3 égale length (2 5 7), mais normalement le changement n'est pas réversible:

- Vous ne pouvez pas faire retourner la dentifrice dans le tube !
- Système d'exploitation:
  - Un fichier supprimer ne peut pas être restauré,
  - Un email envoyé ne peut pas être révoqué,
  - Une page imprimée ne peut pas être 'non imprimée',
- Jeux : un mauvais mouvement dans le jeu d'échec ne peut pas être annulé, …

Logique équationnelle avec symétrie ne semble pas une bonne solutions pour un tel système

PETIT EXEMPLE : VIE

Pour simuler la vie d'une personne :

- Constructeur:

  op person : String Nat - > Person [ctor] .

- Un état type : person (‘’ Walid ’’, 25)
- … cet état doit être changé à person (‘’ Walid ’’, 26)
- **Comment peut on formuler un système comme celui-là ?**

  eq person (X, N) = person(X, N+1) ?

- Résultats dans … |-- person(‘’Walid’’, 40) = person(‘’Walid’’, 40)

LOGIQUE DE RÉÉCRITURE

Logique de réécriture : une logique pour modéliser des / raisonner sur des système dynamique et distribuées / systèmes parallèles

- Système: composé d'une partie statique (type de données etc.) et partie dynamique.
- Partie statique modélisée par des équations,
- Partie dynamique modélisée par des **règles de réécriture**
  - Règles sont utilisées de gauche à droite.
  - Règle : rl person(X , N) => person(X, N+1) .
  - Etiquette : note quelle action est modélisée par la règle.

    rl [brithday] person(X , N) => person(X, N+1) .

SPÉCIFICATION EN LOGIQUE DE RÉÉCRITURE (I)

Supposant La règle :

crl [birthday] : person(X, N) => person(X, N + 1) if N < 1000 .

Signifiée que :

person(X, N) ⟶ person(X, N + 1) est un changement atomique.

SPÉCIFICATION EN LOGIQUE DE RÉÉCRITURE (II)

Définition

$\mathcal{R} = (S, \Sigma, E, L, R)$ est une spécification en logique de réécriture

- (S, Σ, E) : une spécification équationnelle (evtl, sous-sortes)
- L: Ensemble d'étiquettes
- R: Un ensemble de règles de réécriture $l : t \longrightarrow u$ et des règles de réécriture conditionnelles $l : t \longrightarrow u$ **if** $cond$, Où : Cond est une conjuction de testes de qualité et de réécritures.

- Syntaxe d'une règle de réécriture en Maude:

  rl [l] : *t* => *U* .

  crl [l] : *t* => *U if cond* .

- Utiliser mod et endm au lieu de fmod et endfm

SIGNIFICATIONS INTUITIVES

- Un terme ≈ un état d'un système
- Une règles ≈ un changement locale atomique d'un état,
- La séquence

$$R \vdash t \rightarrow u$$

Signifie « qu'il est possible d'atteindre un état u a partir de l'état t » dans la spécification R

Ainsi, On peu inférer

... |-- person("Walid", 40) → person("Walid ", 50)

mais, pas

... |-- person("Walid", 40) → person("Walid ", 20)

Simplement, « une logique équationnelle sans symétrie »

RÈGLES D'INFÉRENCE POUR L R

Réflexivité : $\mathcal{R} \vdash t \longrightarrow t$ pour n'importe quel terme t .

Égalité : Si $\mathcal{R} \vdash t \longrightarrow t'$ et $E \vdash t = u$ et $E \vdash t' = u'$
ainsi, $\mathcal{R} \vdash u \longrightarrow u'$

Congruence : si $\mathcal{R} \vdash t_1 \longrightarrow u_1, \quad \ldots, \quad \mathcal{R} \vdash t_n \longrightarrow u_n$, donc
$\mathcal{R} \vdash f(t_1, \ldots, t_n) \longrightarrow f(u_1, \ldots, u_n)$ pour
n'importe quel symbole de fonction.

Instanciation de règle : pour n'importe quelle règle :

$l : \; t(x_1, \ldots, x_n) \longrightarrow u(x_1, \ldots, x_n)$: Si
$\mathcal{R} \vdash t_1 \longrightarrow u_1, \quad \ldots, \quad \mathcal{R} \vdash t_n \longrightarrow u_n$, Donc
$\mathcal{R} \vdash t(t_1/x_1, \ldots, t_n/x_n) \longrightarrow u(u_1/x_1, \ldots, u_n/x_n)$

Transitivité : Si $\mathcal{R} \vdash t_1 \longrightarrow t_2$ et $\mathcal{R} \vdash t_2 \longrightarrow t_3$, Donc
$\mathcal{R} \vdash t_1 \longrightarrow t_3$

EXÉCUTION DE RÈGLE

- Maude toujours réduit à une forme E-normal en utilisant les équations Avant et Après chaque application de règle.

- Maude suppose que la spécification avec l'ensemble d'équations E est confluente (l'ordre de l'application des règles n'influence pas le résultat) et toujours termine (Elle ne contient pas une boucle infinie),

- La partie gauche d'une règle doit être toujours un terme constructeur,

EXÉCUTION DE RÈGLE

- Spécification de réécriture : souvent la spécification n'est pas confluente et contient des boucle infinies (pas de terminaison).

- Comment peut on exécuter une spécification de réécriture ?
  1. Rewriting (réécritures) : simulation d'un comportement possible.
  2. Searching (rechercher) et Model-cheking pour analyser tous les comportements possibles à partir d'un état initial,

SIMULATION PAR RÉÉCRITURE

- La commande maude rew choisit les règles d'une façon (pseudo-) Aléatoire, Elle continue l'exécutent des règles tant qu'elles sont applicables,
- La commande maude rew [n] exécute au max n pas de réécriture,
- La commande frew similaire à la précédente (mais l'application des équation n'est pas équitable)
- Les termes sont réduits à une forme E-normal avant et après chaque application de règle,
- rew/frew analyse uniquement un seule (à partir de plusieurs) comportement possibles en commençant d'un état de départ.
  - Plusieurs d'autres comportements intéressants ne se voient pas, avec (F)RAW
- Toutefois, on peut rechercher sur tous les comportements possibles à partir d'un état de départ donné.

EXEMPLE : FOOTBALL EUROPÉEN

- Modéliser le Jeu de football,

```
mod GAME is protecting NAT .
   protecting STRING .
   sort Game .
   op _-_ _:_ : String String Nat Nat -> Game [ctor]
   .
   vars HOME AWAY : String .
   vars M N : Nat .
   rl [home-goal] :
      HOME - AWAY M : N => HOME - AWAY M + 1 : N .
   rl [away-goal] :
      HOME - AWAY M : N => HOME - AWAY M : N + 1 .
endm
```

EXEMPLE : FOOTBALL EUROPÉEN

```
Maude> rew [5] "MC Alger" - "ES Sétif" 0 : 0 .
result Game: "MC Alger" - "ES Sétif" 3 : 2

Maude> rew [1] "WA Tlemcen" - " ASO Chlef " 0 : 0 .
result Game: "WA Tlemcen" - " ASO Chlef " 1 : 0
```

Tout va bien, mais . . .

```
Maude> rew [5] "Lybie" - "Tunis" 0 : 0 .
result Game: "Lybie" - "Tunis" 3 : 2 .
```

Mieux avec \fair" rewriting (frew)?

```
Maude> frew [5] "Lybie" - "Tunis" 0 : 0 .
result Game: "Lybie" - "Tunis" 3 : 2 .
```

Observation: Apparemment les visiteurs ne gagnent jamais un match!

EXEMPLE : VIE D'UNE PERSONNE (I)

- Modéliser une version simple de la vie d'une personne,

```
mod ONE-PERSON is
   protecting NAT .
   protecting STRING .
   sorts Person Status .
   op person : String Nat Status -> Person [ctor] .
   ops single engaged married separated divorced
       deceased widow widower : -> Status [ctor] .

   var X : String . var N : Nat . var S : Status .

   crl [birth-day] :
      person(X, N, S) => person(X, N + 1, S)
         if N <= 1000 /\ S =/= deceased .
```

EXEMPLE : VIE D'UNE PERSONNE (II)

```
    crl [successful-proposal] :
          person(X, N, S) => person(X, N, engaged)
       if N >= 15 /\ (S == single or S == divorced) .

    rl [marriage] : person(X, N, engaged) =>
                    person(X, N, married) .

--- Rules for death, separation, broken-engagement,
--- divorce, death of spouse (?) should be added!!

endm
```

SIMULATION D'UNE PERSONNE

```
Maude> rew [10] person("Walid SAHLI", 63, married) .
result Person: person(" Walid SAHLI", 73, married)
```

```
Maude> frew [5] person("Noam Chomsky", 81, married) .
result Person: person("Noam Chomsky", 86, married)
```

OK, Est ce qu'une personne peut devenir plus jeune ?

RECHERCHE DANS MAUDE (I)

Rechercher dans tous les états (termes) qui peuvent être atteint à partir d'un état donné.

search startterm arrow pattern [such that cond] .

Où arrow est l'un de :

=>1 (atteignable dans un seul pas)
=>* (atteignable dans 0 ou plusieurs pas)
=>+ (atteignable dans 1ou plusieurs pas)
=>! ("Etat final")

et pattern est un terme (possible, avec des variables) et cond est une condition de réécriture

- Limiter le nombre de résultats : serach [n] ….
- … et limiter aussi le nombre de pas de réécriture : search [n,d]…

RECHERCHE DANS MAUDE (II)

- La rechercher dans maude est de type « parcours_en_largeur » « breadth-first search » dans l'arbre de calcul « computation tree » à partir d'un terme.
  - Si on veut n solutions, et ces solutions existent, Maude va les trouver.
  - Si n solutions n'existent pas, et le nombre d'états atteignables à partir l'état initial est infini, ainsi la rechercher ne va pas se terminer,
- Le pattern de recherche doit être un terme constructeur.
- Utiliser des variables dans le terme de recherche soit de forme var:sorte, ou utiliser les variables déclarées dans le module.
- Le résultat sera une substitution à partir du terme jusqu'au l'état trouvé.
- Show path n . Imprime le chemin à l'état numéro n dans le recherche précédent,

EXEMPLE : RECHERCHER DANS FOOTBALL (I)

- Est-ce que l'équipe visiteur peut gagner le match de foot?

```
Maude> search [2]
       "MC Alger" - "ES Sétif" 0 : 0 =>*
       "MC Alger" - "ES Sétif" M:Nat : N:Nat
       such that M:Nat + 5 < N:Nat .

Solution 1 (state 27)
M --> 0
N --> 6

Solution 2 (state 35)
M --> 0
N --> 7.
```

EXEMPLE : RECHERCHER DANS FOOTBALL (II)

- Imprimer le chemin au résultat 0-7 :

```
Maude> show path 35 .

state 0, Game: "MC Alger" - "ES Sétif" 0 : 0
===[ rl HOME - AWAY M : N =>
         HOME - AWAY M : N + 1 [label away-goal] .
       ]===>
state 2, Game: "MC Alger" - "ES Sétif" 0 : 1
===[ rl ... [label away-goal] . ]===>
   ...
===[ rl ... [label away-goal] . ]===>
state 27, Game: "MC Alger" - "ES Sétif" 0 : 6
===[ rl ... [label away-goal] . ]===>
state 35, Game: "MC Alger" - "ES Sétif" 0 : 7
Maude>
```

EXEMPLE : RECHERCHER DANS FOOTBALL (III)

- Et si on a besoin uniquement à la liste des nom des règles du chemin au terme 35, on peut utiliser la commande show path labels n :

```
Maude> show path labels 35 .
    away-goal
    away-goal
    away-goal
    away-goal
    away-goal
    away-goal
    away-goal
Maude>
```

EXEMPLE : RECHERCHER DANS FOOTBALL (IV)

- Si l'équipe visiteur gagne, est ce que l'arbitre peu annuler le but ?

```
Maude> search [1] "Marroc" - "Tunis" 0 : 1 =>*
"Marroc" - "Tunis" 0 : 0 .
```

Le système ne terminera jamais mais ça ne réfute pas l'hypothèse!

Maintenant, quand est ce que cette recherche se termine ?

```
Maude> search person("Walid SAHLI", 40, single) =>!
       P:Person .

Solution 1 (state 2900)
P:Person --> person("Walid SAHLI", 1001, married)

No more solutions.
```

LA RECHERCHE EN MAUDE

- La recherche utilisée est la recherche par parcours-en-largeur « breadth-first » à travers l'arbre de tous les comportements possible à partir d'un état initial .
- Efficacité du Maude : sauvegarde tous les états visités !
  - Ne continue pas la recherche à partir d'un état déjà visité.
- Problème d'espace mémoire: tous les états visités doivent être sauvegardés dans la mémoire centrale,
- Maude fait le travail efficacement, toutefois le problème est avec le nombre d'états qui ne terminent pas,
  - Ce que reflet la complexité du problème. (mémoire/ temps ..etc.)
  - Le nombre d'états visités avec n pas de calcul est presque x^n si à partir de chaque état, X états est atteignable après un seul pas de réécriture.