Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département d'Informatique
2ème année Master
Option: Génie Logiciel et Systèmes distribués
Module : Logique de Réécriture et ses Applications

Introduction

Présenté par : Dr. Mohamed RAMDANI

Nb: Une grande partie du matériel présenté dans ces diapositives fait partie du cours INF3230 de l'université d'Oslo.

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département d'Informatique
2ème année Master
Option: Génie Logiciel et Systèmes distribués
Module : Logique de Réécriture et ses Applications

Information sur le cours LRA

- ***Faculté*** des sciences exactes et des sciences de la nature et de la vie
- ***Département :*** Informatique
- ***Intitulé du cours*** *:* Logique de Réécriture et ses Applications
- ***Crédit*** *:* 06
- ***Coefficient :*** 03
- ***Durée :*** 14 semaines
- ***Horaire :*** Lundi 9h30- 12h40
- ***Place :*** *Salle 2*
- ***Enseignant :*** Cours et TP : Dr. Mohamed RAMDANI.
- ***Mail :*** mohamed.ramdani@univ-biskra.dz

- **Disponibilité :** Au bureau, laboratoire numéro 07 : Lundi et mardi.

Sur le forum et par mail je m'engage à répondre aux questions relatives au cours du mieux que je peux.

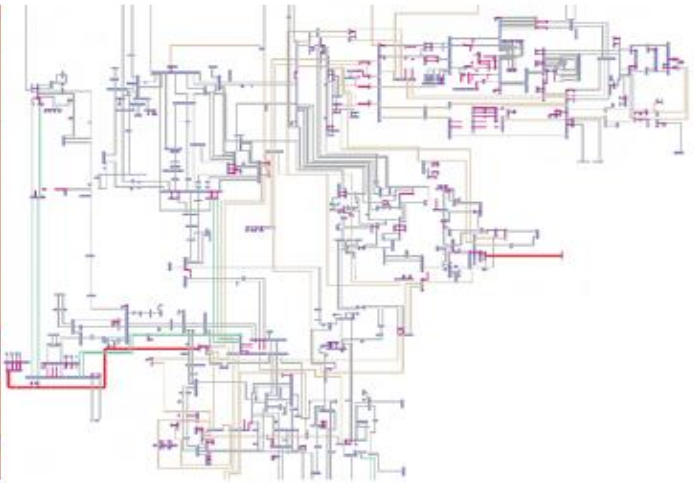
Systèmes distribués

Aujourd'hui, La plupart des systèmes informatiques sont des systèmes distribués avec plusieurs « ordinateurs » qui communiquent entre eux afin d'atteindre des objectifs bien définis, e.g:

- E-banking,
- Systèmes de réservation aériennes,
- Système de gestion de bases de données répartis,
- Avions et applications aéronautiques
- Applications militaires modernes.
- Web services

Motivation

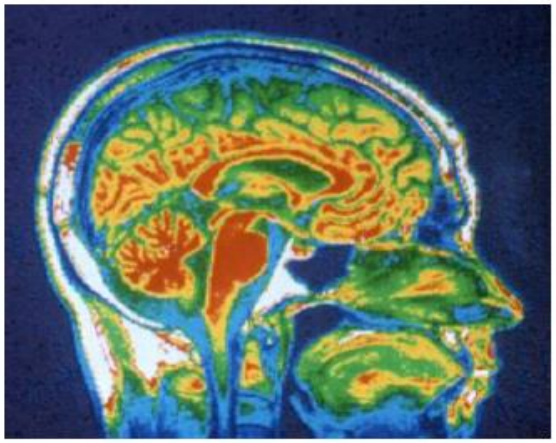
La société moderne dépend sur des systèmes informatiques avancés, qui sont :



- Complexes , Larges et aussi Critiques
- Très difficile à comprendre.


Nordea
exchange
Nordea
cash


A380
A380




KLM
Plan & book your flight
Christmas offers
More offers
Journeys of inspiration


FINN
Mulighetenes marked
Næringsmarked
Møteplassen
Torget
Tjenester
Jobb
Eiendom
Kart
Sett inn annonse
Båt
Bil
MC
Reise
235303 annonser på FINN. 8217 nye i dag.

Motivation

- Taille + Complexité+ Criticité / couts :

- Il faut comprendre très bien le (problème / système) avant de l'implementer !

Généralement, les programmes contiennent des erreurs dans leurs première version,

Toutefois …

- On ne peut pas tester un programme de contrôle d'avion sur Airbus A380,
- Ou tester un protocole de commerce électronique met en action sur internet.

- Donc, il est plus pratique d'analyser les systèmes avant des les implémenter.

Modélisation

- Créer un model (une abstraction) du système,

  puis l'analyser le modèle et l'implementer par la suite.

Modélisation : Buildings

- Ne jamais commencer la construction par une idée approximative de ce que vous voulez.

Modélisation : Buildings

- Ne jamais commencer la construction par une idée approximative de ce que vous voulez.

Modélisation : Buildings

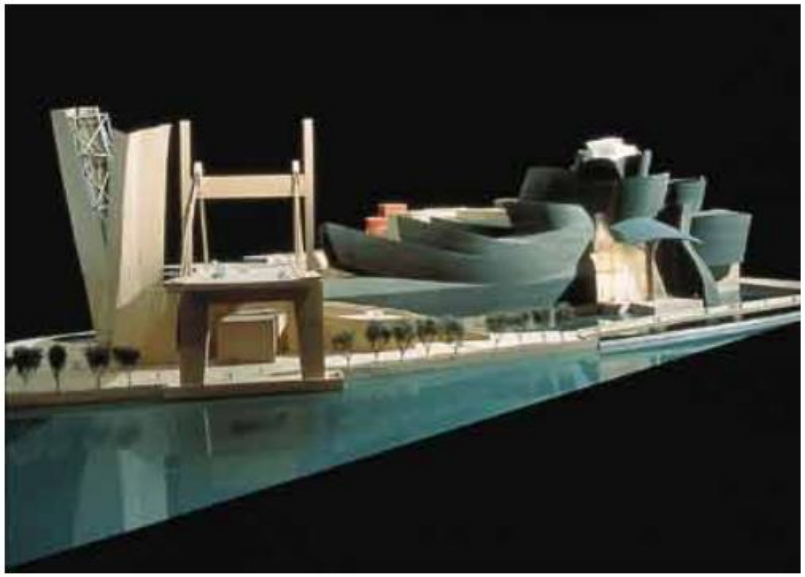
- Ne jamais commencer la construction par une idée approximative de ce que vous voulez.

Modélisation : Buildings

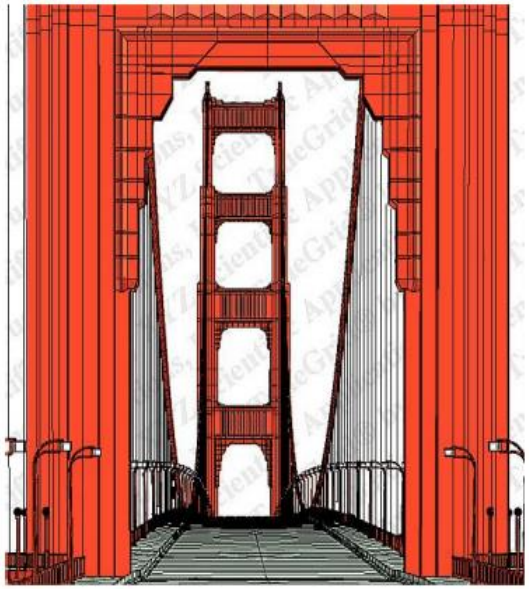
- Ne jamais commencer la construction par une idée approximative de ce que vous voulez.

Modélisation : Buildings

- Plutôt; il faut développer un modèle du building avant de le construire .

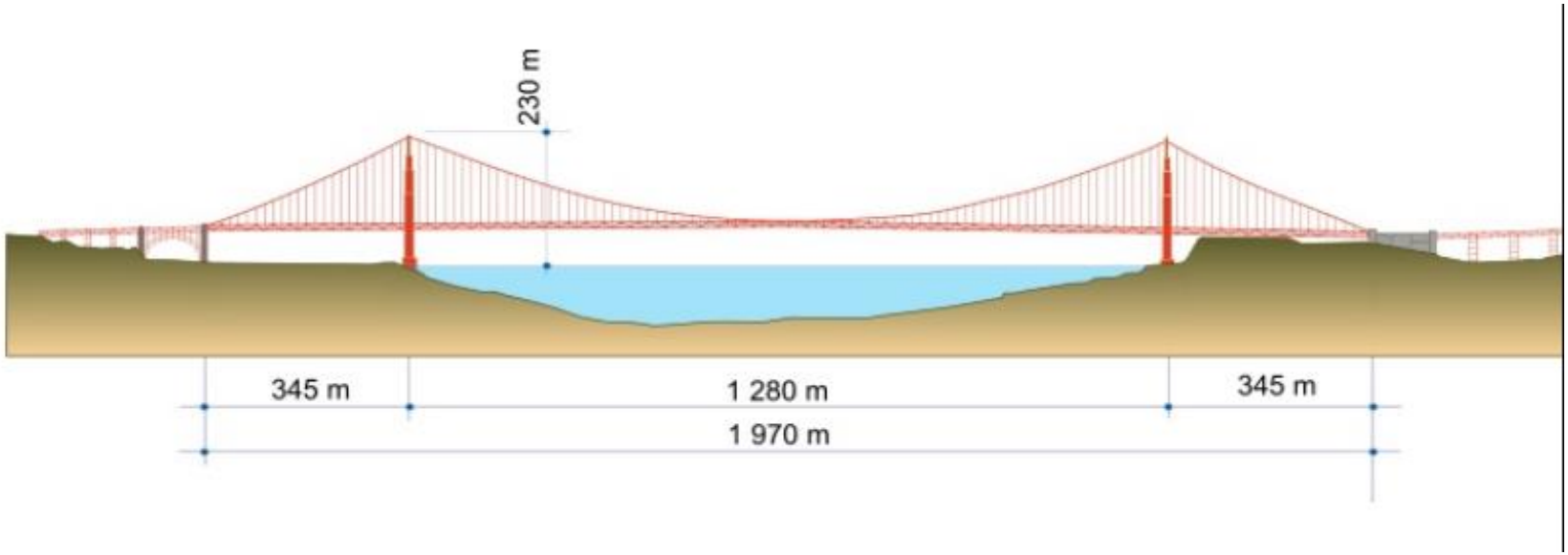


Modélisation : Buildings



Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .



Modélisation : Buildings

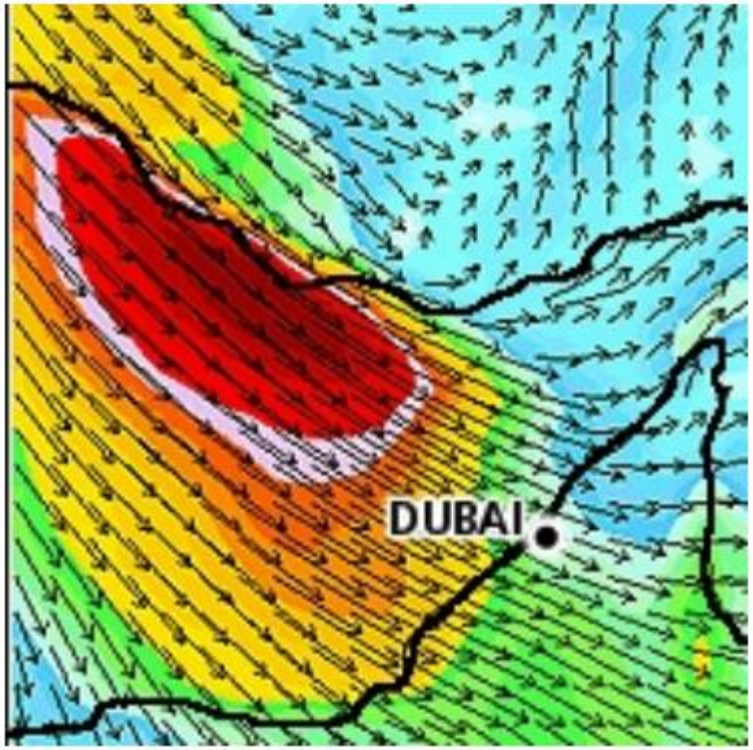
- Plutôt développer un modèle du building avant de le construire .

Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .

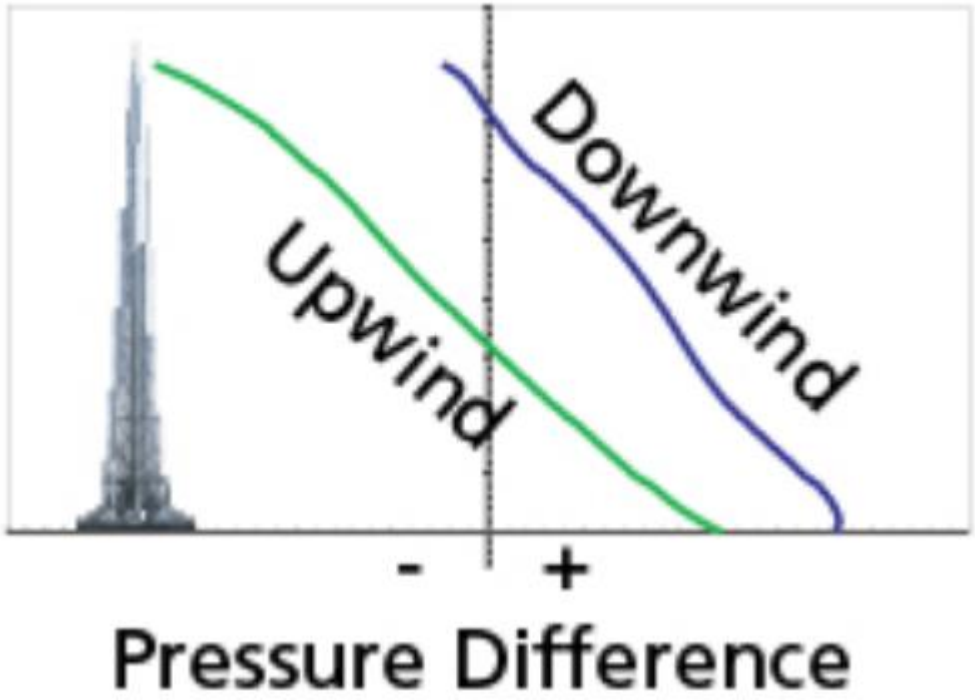
Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .



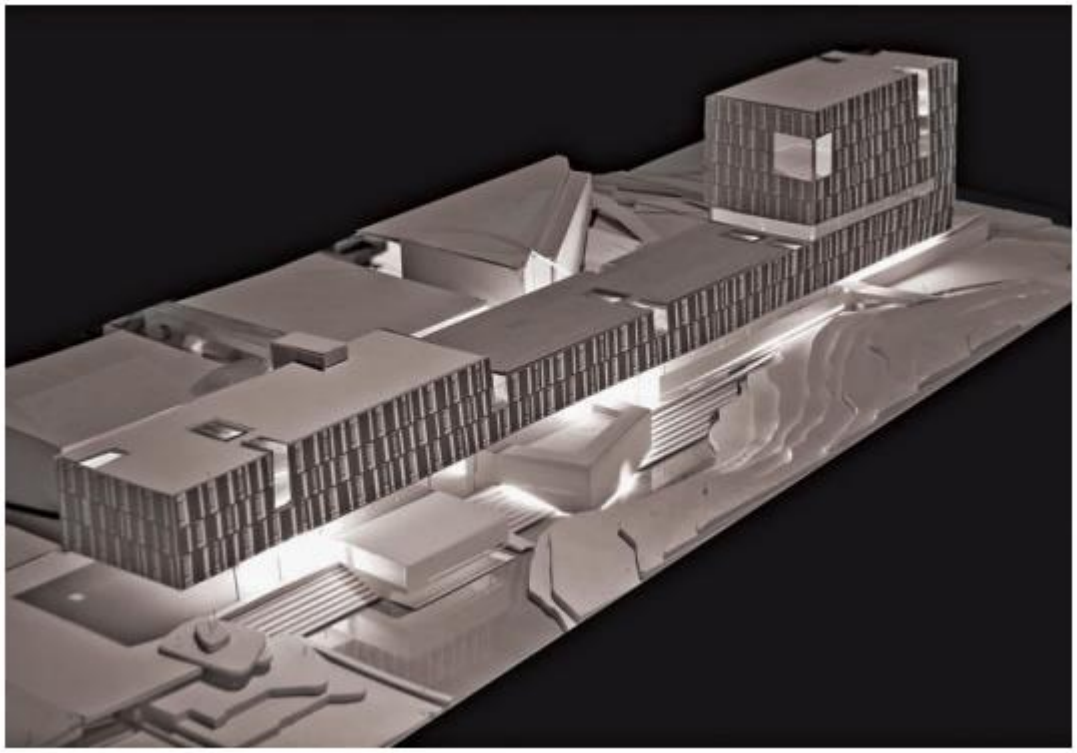
Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .



Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .

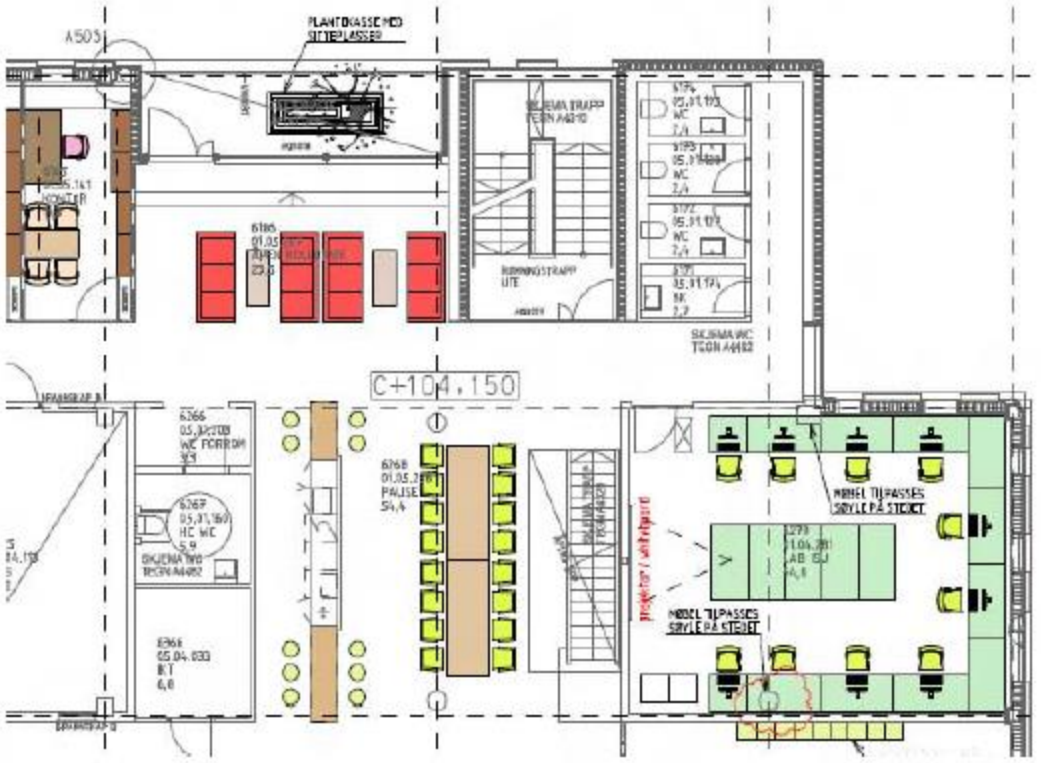


Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .

Modélisation : Buildings

- Plutôt développer un modèle du building avant de le construire .



Avantages de modélisation des Buildings

- C’est facile et rapide de construire un modèle.
- Facile de le tester avec d’autres idées de conceptions

- Analyser la conception
- Esthétique
- Plan/ disposition
- Fonctionnalité : Stabilité, Vent, tremblements de terre, courants, ….
- ……
- Simple de construire des buildings a partir des modèles/ dessins

Modélisation des Systèmes Informatiques

- Besoin à des modèles (conceptions) des systèmes informatiques
- Il faut que ces modèles soit Rapide et Facile à construire un modèle :
- Haut niveau d'abstraction : faire abstraction de tout les détailles non nécessaires :

- Exemple : comment-il doit réagir mon protocole de communication si un message n'atteindre pas sa destination ?
  - Le contenu du message n'est pas important pour le protocole
  - L'implémentation du mécanisme du transport de messages entre deux ordinateurs n'est pas aussi important pour le protocole
  - …… le modèle doit ignorer ces détailles

- Analyser la conception !
  - « Regardant » le modèle (UML)
  - Analyser le modèle par ordinateur !
  - Exécuter le modèle : testes, simulation, autres types d'analyse (« model-checking»)

- Finalement: vérifier le modèle.

Modélisation Formelle

Un modèle formel définit un objet mathématique

- Précis, non ambiguë .

On peut analyser le modèle (mathématiquement) soit :

- Par main.
- Par ordinateur.

Ce Module

- Concepts de la logique de réécriture
- Modélisation de haut-niveau des systems distribués avec Maude
- Programmation fonctionnelle de haut niveau
- Analyse des modèle avec des specifications exécutables sur ordinateur.
- Compréhension des systèmes distribués.
- Bases de données distribuées
- Protocoles de communication
- Protocole de Cryptographie (sécurité)
- Autres type de système de communication.
- Compréhension de haut niveau de différent types de communication
- Synchrone (« handshake») Vs. asynchrone
- Unicast / multicast/ broadcaste
- Ordonnée (FIFO) Vs communication non ordonnée
- Autres type de système de communication
- Pas de détaille d'implémentation !

Ce Module (2)

- Différent types pour analyser les systèmes distribués
- Simuler un seul comportement .
- Analyse exhaustive avec le model-checking pour trouver des erreurs.

Logique de Réécriture (1)

- Étudiée depuis les années 80s.
- Une plateforme logique dans laquelle d'autres logiques peuvent être représentées.
- Une plateforme sémantique pour la spécification des langages et des systèmes.
- Basée sur des simples règles de déductions.

Logique de Réécriture (2)

- Les axiomes de base sont des règles de réécriture
- *t ->t'* où t et t' sont des expressions dans un langage donné.
- C'est la logique de concurrence et de changement.

Logique de Réécriture (3)

- Deux lectures complémentaires pour les règles de réécriture : t->t’

- Calcul (Computational)

  Transition locale dans un système concurrent.
  t et t’ décrivent des patternes d’un sous état distribué d’un système.
  La règle explique comment une transition locale peut avoir lieu.

- Logique (Logical)

  La règle de réécriture t->t’ est interprétée comme une règle d’inférence, ainsi on peut déduire la formule t’ a partir de celle de t.

Logique de Réécriture (4)

Elements de Calcul, Réécriture, logique

- Les points de vues ‘calcul et logique’ ne sont pas exclusifs, ils se sont complémentaires.

State	$\leftrightarrow$ *Term*	$\leftrightarrow$ *Proposition*
Transition	$\leftrightarrow$ *Rewriting*	$\leftrightarrow$ *Deduction*
Distibruted Structure	$\leftrightarrow$ *Algebraic Structure*	$\leftrightarrow$ *Propositional Structure*

Logique de Réécriture : Définitions

- Une signature dans la LR est une théorie équationnelle (Σ, E) où Σ est une signature équationnelle et E est un ensemble de Σ -equations.

- La réécriture s'effectuée sur des classes d'équivalance de termes modulo E

- Soit une signature (Σ, E), les expression de la logique de réécriture sont des sequences de la forme:
- où t et t' sont des Σ-terms qui contiennent éventuellement des variables et $[t]_E$ denote la classe d'équivalance du term t modulo les équations E
- Termes
- Variables, Constantes, Opérateurs

Théorie de Réécriture

- 4-tuple: (Σ, E, L, R)
- (Σ, E) : Théorie équationelle
- Σ : signature équationnelle
- E: un ensemble de Σ-equations
- L: un ensemble d'etiquettes (labels)
- R: un ensemble de règles étiquetées (peuvent être conditionnelles)

Théorie de Réécriture: Règle de Déduction

(*I*) **Reflexivity**. For each $[t] \in T_{\Sigma,E}(X)$, $\dfrac{}{[t] \longrightarrow [t]}$.

(*C*) **Congruence**. For each $f \in \Sigma_n$, $n \in \mathbb{N}$,

$$\frac{[t_1] \longrightarrow [t'_1] \quad \ldots \quad [t_n] \longrightarrow [t'_n]}{[f(t_1, \ldots, t_n)] \longrightarrow [f(t'_1, \ldots, t'_n)]}.$$

(*R*) **Replacement**. For each rule $l : [t(x_1, \ldots, x_n)] \longrightarrow [t'(x_1, \ldots, x_n)]$ in R,

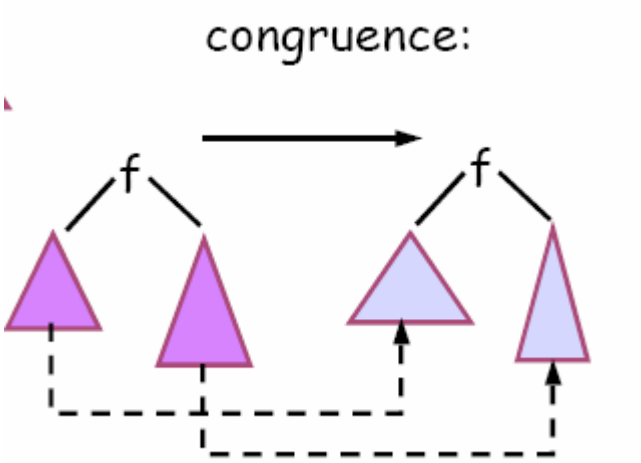
$$\frac{[w_1] \longrightarrow [w'_1] \quad \ldots \quad [w_n] \longrightarrow [w'_n]}{[t(\overline{w}/\overline{x})] \longrightarrow [t'(\overline{w'}/\overline{x})]}.$$

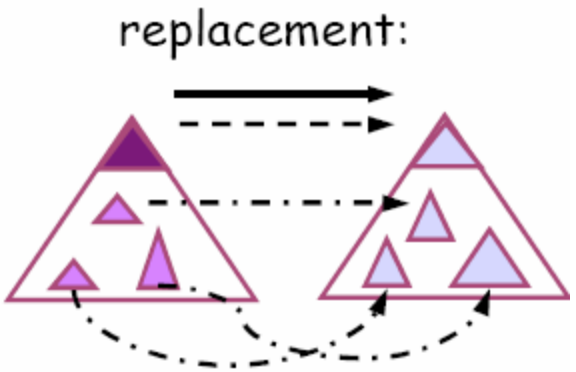
(*T*) **Transitivity**

$$\frac{[t_1] \longrightarrow [t_2] \qquad [t_2] \longrightarrow [t_3]}{[t_1] \longrightarrow [t_3]}.$$

Théorie de Réécriture: Règle de Déduction

closed under





Règles dérivées

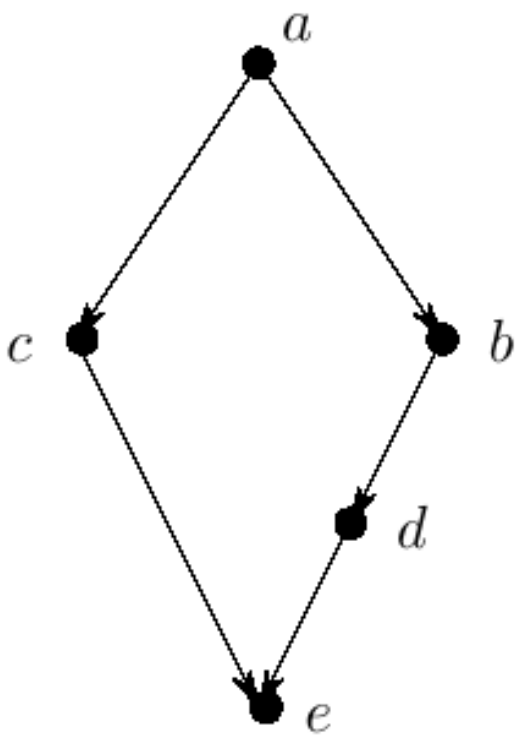
- *Unconditional replacement:* For each rule $\dfrac{}{[t(\bar{x})] \to [t'(\bar{x})]} \in R$,

$$\frac{[u_1] \to [u'_1] \quad \ldots \quad [u_n] \to [u'_n]}{[t(\bar{x}/\bar{u})] \to [t'(\bar{x}/\bar{u'})]}$$

- *Substitution:*

$$\frac{[t(\bar{x})] \to [t'(\bar{x})] \quad [u_1] \to [u'_1] \quad \ldots \quad [u_n] \to [u'_n]}{[t(\bar{x}/\bar{u})] \to [t'(\bar{x}/\bar{u'})]}$$

Example I: Systèmes de Transition



$$\Sigma = \{a, b, c, d, e\}$$

$$E = \emptyset$$

$$\begin{array}{lll} R & : & a \to b \\ & & a \to c \\ & & b \to d \\ & & d \to e \\ & & c \to e \end{array}$$

$$\dfrac{\dfrac{}{[a] \to [c]}\ (\mathrm{R}) \qquad \dfrac{}{[c] \to [e]}\ (\mathrm{R})}{[a] \to [e]}\ (\mathrm{T})$$

Références

1.M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Martı-Oliet, J. Meseguer, and C. L. Talcott. All About Maude — A High-Performance Logical Framework.How to Specify, Program and Verify Systems in Rewriting Logic, volume 4350 of Lecture Notes in Computer Science. Springer, 2007

1.N. Mart´ı-Oliet and J. Meseguer. Rewriting logic as a logical and semantic framework. In D.Gabbay, editor, Handbook of Philosophical Logic. Second Edition, volume 9, pages 1–81. Kluwer Academic Press, 2002.

1.Maude Manual.

1.P. Olveczky **" Formal Modeling and Analysis of Distributed Systems in Maude"**, University of Oslo, Norway.

1.N.A. Harman **" System specification course"**, Swansea university, UK.

1.The Maude System Webage : http://maude.cs.uiuc.edu/

1.Clavel, M., **Durán**, F., **Eker**, S., **Lincoln**, P., **Martí-Oliet**, N.,**Meseguer**, J., **Talcott**, C. « All About Maude - A High-Performance Logical Framework", Springer