Que peut faire Coq pour les systèmes de base de données? (Travail en cours)

Yoichi Hirai Reynald Affeldt

AIST

« SELECT * FROM menu ; »

Cette présentation

Analyse syntaxique

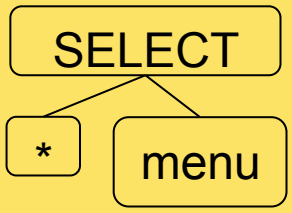
Normalisation

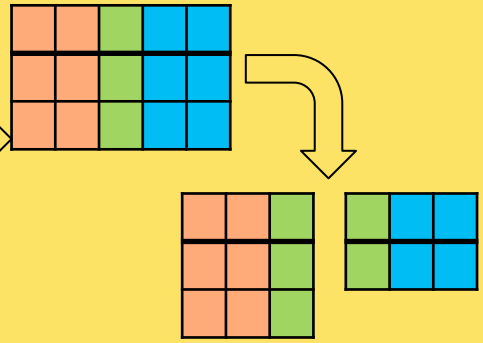
Optimisation des requêtes

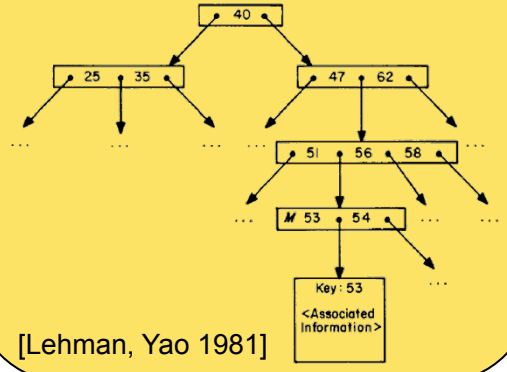
Arbre-indexe

Concurrence
La récupération d'un crash

Base de données







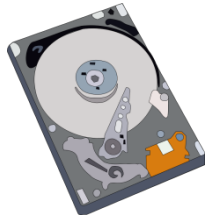
[Lehman, Yao 1981]

Fig. 2. An example B*-tree (with parameter $k = 2$).

Couche du langage de manipulation

Couche du modèle relationnel

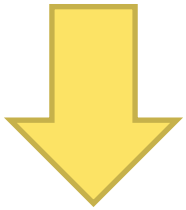
Couche des disques et de la mémoire



Les *anomalies* : Problèmes de cohérence

- Un problème d'écriture

Identificateur du Titre	Titre	Auteur	Bibliothèque
3	Istanbul	Orhan Pamuk	Centrale
3	Istanbul	Orhan Pamuk	Est



On change le titre, mais on échoue à changer toutes les occurrences ⇒ La cohérence est brisée

Identificateur du Titre	Titre	Auteur	Bibliothèque
3	Istanbul	Orhan Pamuk	Centrale
3	Mon nom est Rouge	Orhan Pamuk	Est

II <>

La première forme normale

La 1ère forme normale interdit la duplication du même attribut.

Identificateur	Titre	Auteur	Bibliothèque	Bibliothèque
3	Istanbul	Orhan Pamuk	Est	Centrale
5	Les Thibault	Roger Martin du Gard	Centrale	Ouest

Même semantique mais syntaxe différente

Identificateur	Titre	Auteur	Bibliothèque	Bibliothèque
3	Istanbul	Orhan Pamuk	Centrale	Est
5	Les Thibault	Roger Martin du Gard	Centrale	Ouest

Dépendance Fonctionnelle

Identificateur du Titre	Titre	Auteur	Bibliothèque
3	Istanbul	Orhan Pamuk	Centrale
3	Istanbul	Orhan Pamuk	Ouest

{Identificateur du titre} → {Titre, Auteur}
{Identificateur du titre, Bibliothèque} →
{Identificateur du titre, Titre, Auteur, Bilbliothèque}

La deuxième forme normale

interdit cette table :

profID	Nom du professeur	En poste depuis	Jour	Horaire	Cours
33	R. Wavey	1951-09-01	Mercredi	15:00-	Physique 2A
34	…	…	…	…	…

…en raison de la dépendance partielle :

1. {profID, Jour, Horaire} est un ensemble minimal qui détermine tous les attributs. (Cet ensemble est une *clé*).
2. La date d'entrée en poste n'est pas contenue dans la clé. (C'est un *attribut secondaire*)
3. La date d'entrée en poste dépend d'un ensemble {profID}, qui est un sous-ensemble (stricte) d'une clé {profID, Jour, Horaire}.

La troisième forme normale

interdit cette table:

Tournoi	Année	Vainqueur	Date de naissance du vainqueur
Indiana Invitational	1998	Al Fredrickson	21 Juillet 1975
Des Moines Masters	1999	Al Fredrickson	21 Juillet 1975
Indiana Invitational	1999	Chip Masterson	14 Mars1977

Exemple de en.wikipedia.org

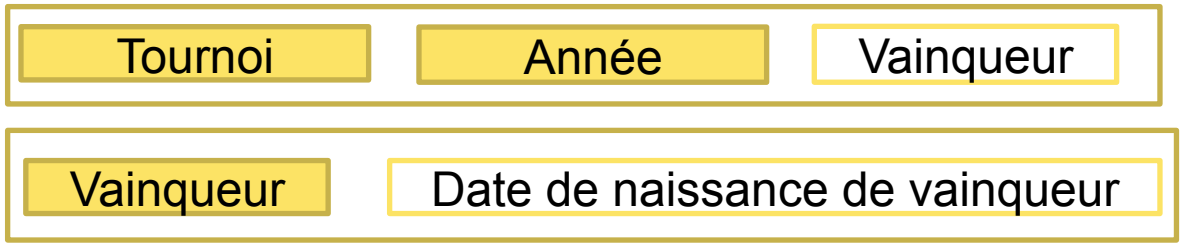
…en raison de la dépendance transitive :

1. “Date de naissance du vainqueur” est un attribut secondaire
2. {Tournoi, Année} est une clé
3. {Tournoi, Année} → {Vainqueur} est vraie
4. {Vainqueur} → {Tournoi, Année} n’est pas vraie
5. {Vainqueur} → {Date de naissance du vainqueur} est vraie
6. “Date de naissance du vainqueur” n’est pas dans {Tournoi, Année}
7. “Date de naissance du vainqueur” n’est pas dans {Vainqueur}

Obtenir la troisième forme normale

- Entrée: un ensemble fini de dépendances fonctionnelles

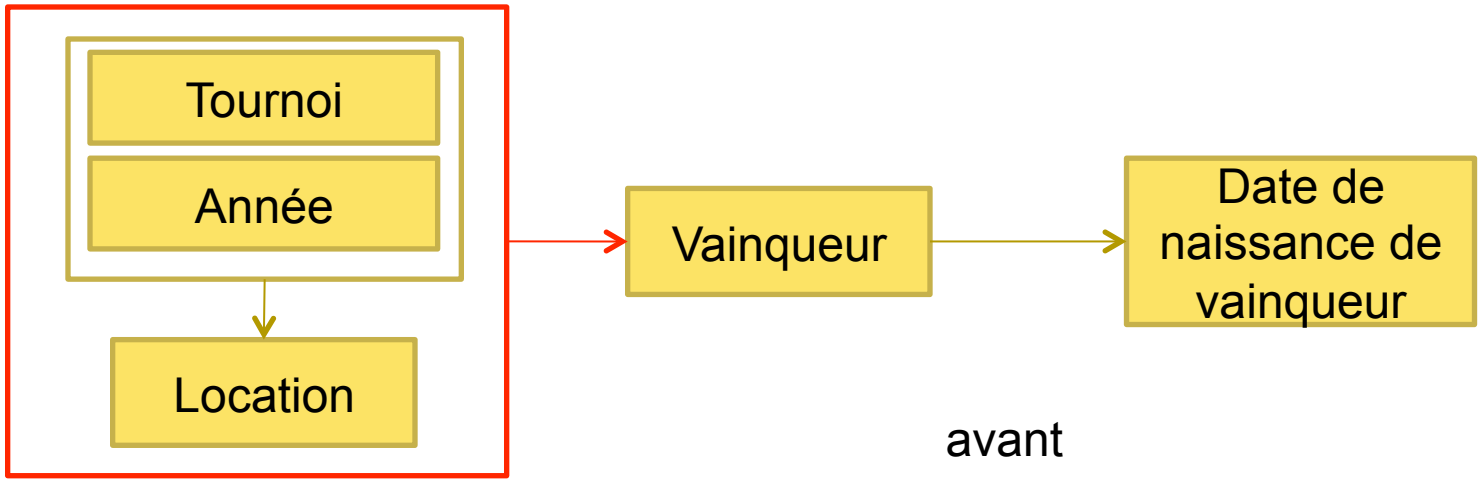
- Sortie: un ensemble fini de relations et leurs clés (en 3FN) qui réalise les dépendances donnée en entrée.

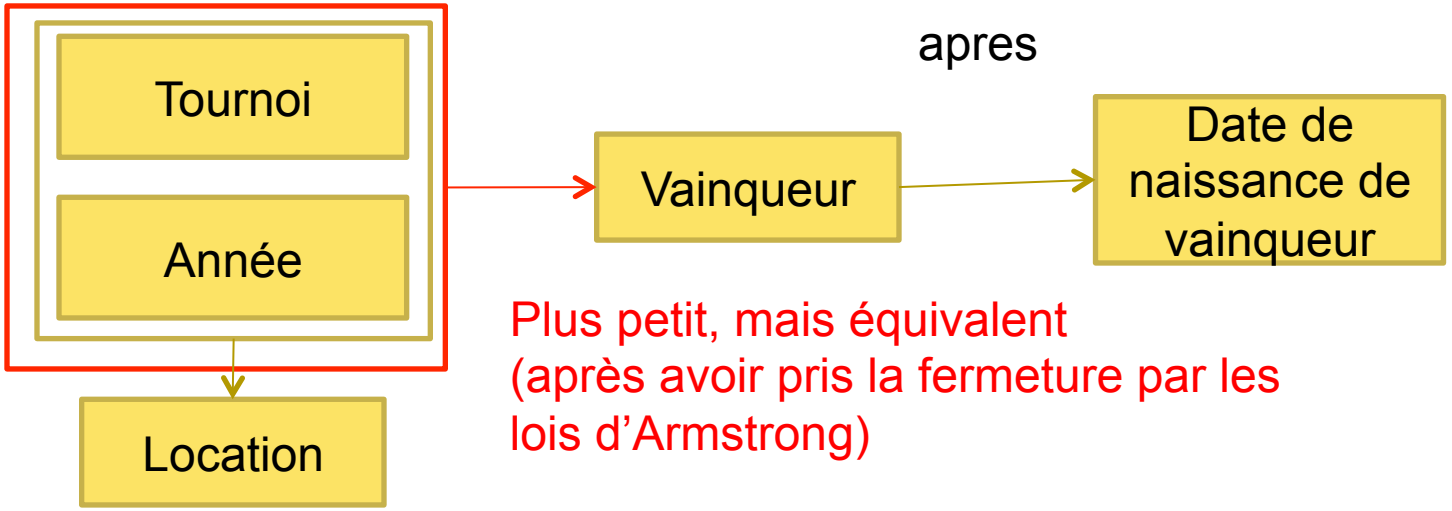


- Histoire de la troisième forme normale :

1972 Codd → définition des formes normales
1972 Delobel et Casey → une tentative erronée
1975 Wang et Wedekind → une tentative erronée
1976 Bernstein → une solution

Étape 1/4 : éliminer les attributs superflus





Plus petit, mais équivalent
(après avoir pris la fermeture par les lois d'Armstrong)

Étape 2/4 : effacer des dépendances inférables

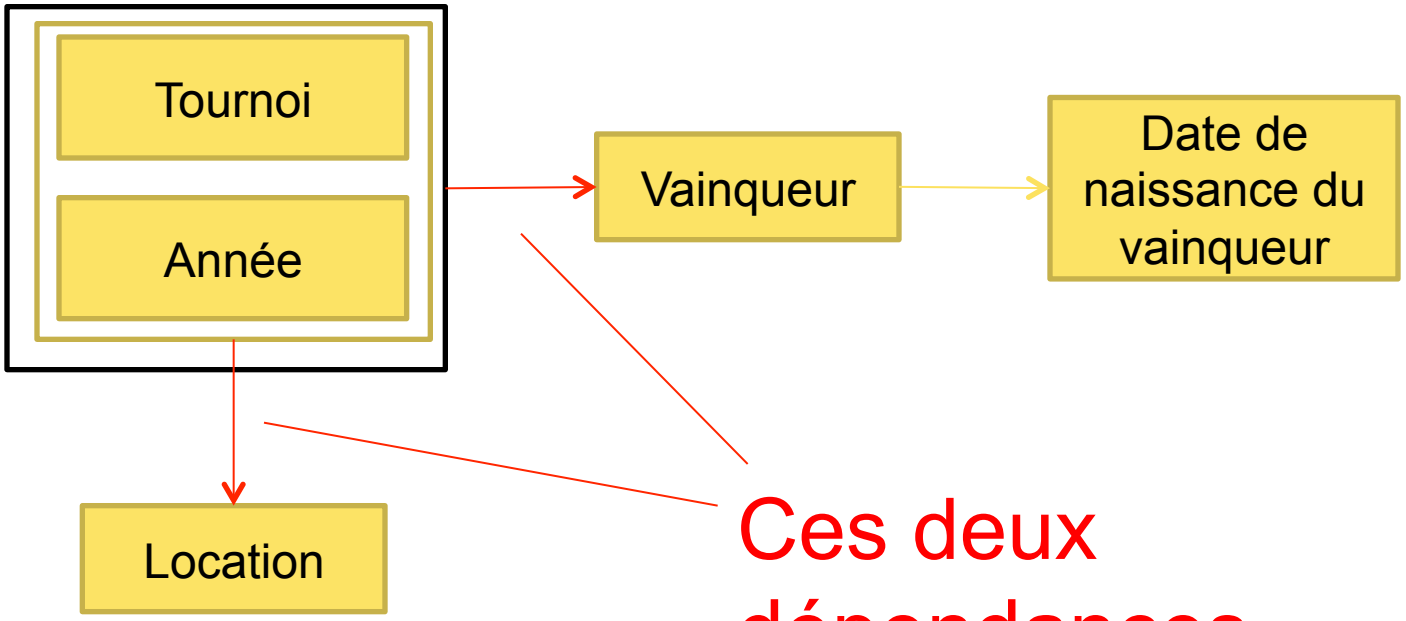
```
Definition find_covering_step fds : seq simple_dep * bool :=
  match find_elm (fun fd => redundant fd fds) fds with
    | None => (fds, true)
    | Some f => (remove' f fds, false)
  end.

Function find_covering fds {measure size fds} :=
  match find_covering_step fds with
    | (fds', true) => fds'
    | (fds', false) => find_covering fds'
  end.
 move => fds fds' _ _ E.
 move: (find_covering_step_spec fds).
 by rewrite E /=; move => [_ ?]; apply/ltP.
Defined.
```

Effacement répété des dépendances inférables

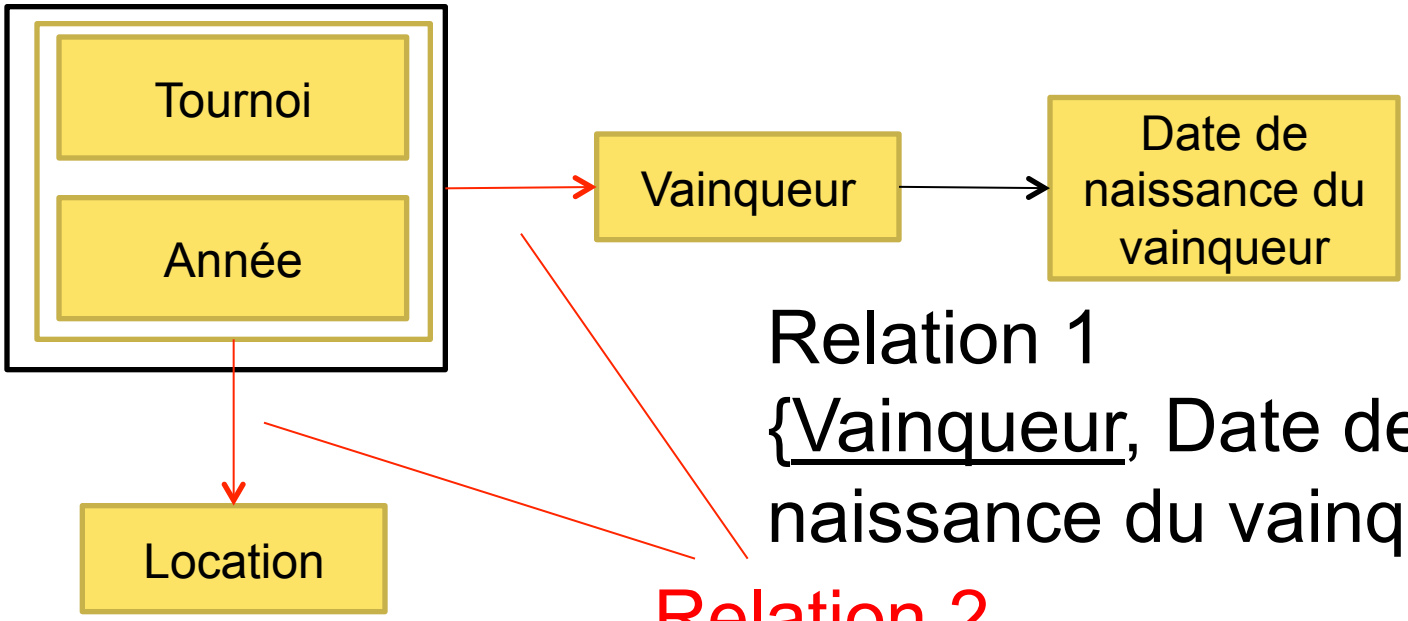
Prouver la termination

Étape 3/4 : partition



Ces deux dépendances fonctionnelles partagent la même partie gauche.

Étape 4/4 : construction des relations



Relation 1
{Vainqueur, Date de naissance du vainqueur}

Relation 2
{Tournoi , Année, Location, Vainqueur}

Les attributs soulignés sont des clés.

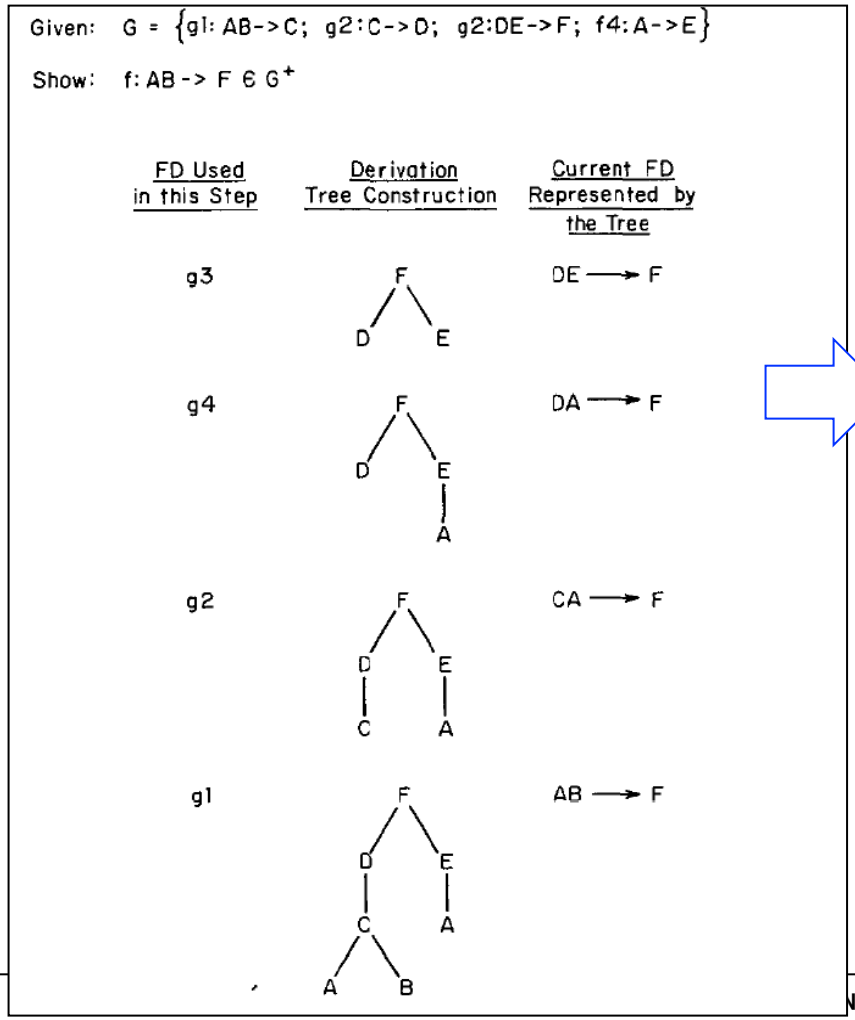
On veut prouver en Coq que ces relations sont en 3FN.

Prouver la mise en 3FN

- Essentiellement en suivant la preuve papier de Bernstein
  21 lines on paper
  199 lines in ssreflect
- Formalisation des structures de donnée qui représentent les inférences (transparent suivant)

Formalisation de la démonstration de Bernstein [1976]

"nodes", "leaf", sans définitions




```
Inductive closure : seq att -> seq att -> Prop :=
| orig: forall X A,
    theory (X, A) -> closure X [:: A]
| perm: forall X X' Y Y',
    X =i X' -> Y =i Y' -> closure X Y
                       -> closure X' Y'
| fd_rfl:
    forall (X Y : seq att), {subset Y <= X}
      -> closure X Y
| fd_aug:
    forall X Y Z W, {subset Z <= W} ->
     closure X Y -> closure (X ++ W) (Y ++ Z)
| fd_trans:
    forall X Y Z, closure X Y -> closure Y Z
      -> closure X Z
.
```

Ce sont les lois d'Armstrong [1974].

Les axiomes d'Armstrong [1974]

1. La réflexivité: $Y \subseteq X$ implique $X \rightarrow Y$
2. L'augmentation: $Z \subseteq W$ et $X \rightarrow Y$ impliquent $X \cup W \rightarrow Y \cup Z$
3. La transitivité: $X \rightarrow Y$ et $Y \rightarrow Z$ impliquent $X \rightarrow Z$

Il s'agit d'une axiomatisation correcte et suffisante de la sémantique relationnelle des dépendances fonctionnelles

La terminaison des algorithmes

- La terminaison de la fermeture des inférences (vis-à-vis des axiomes d'Armstrong) [étape 2/4]
  - On obtient une séquence d'ensembles.
  - Les tailles convergent parce que croissantes et délimitées (nombre fini d'attributs)
  - Quand les tailles convergent, la fermeture est définie
- La terminaison de l'algorithme de Bernstein
  - C'est plus facile parce que toutes les étapes sont une simplification
  - Répéter la simplification jusqu'à ce que c'est impossible de la faire

Aperçu du code Coq

Module	Lignes de code	Contenu
Les axiomes d'Armstrong et leur fermeture	562 + 104	104 lignes pour prouver qu'une séquence délimitée monotone converge.
Les définitions et les propriétés des étapes	630	Preuves un peu répétitives.
L'algorithme produit une 3FN.	199	Technique (beaucoup de variables)

« SELECT * FROM menu ; »

Cette présentation

Analyse syntaxique

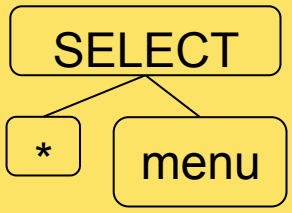
Normalisation

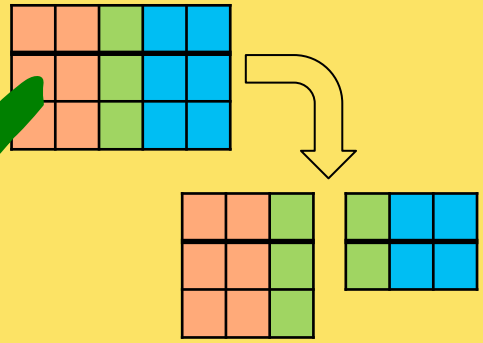
Optimisation des requêtes

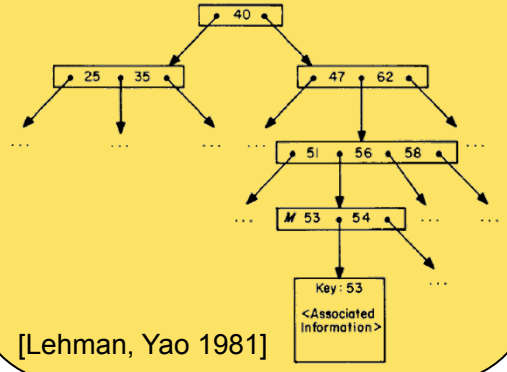
Arbre-indexe

Concurrence
La récupération d'un crash

Base de données







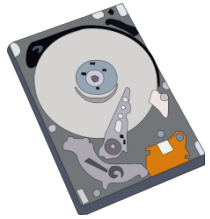
[Lehman, Yao 1981]

Fig. 2. An example B*-tree (with parameter $k = 2$).

Couche du langage de manipulation

Couche du modèle relationnel

Couche des disques et de la mémoire



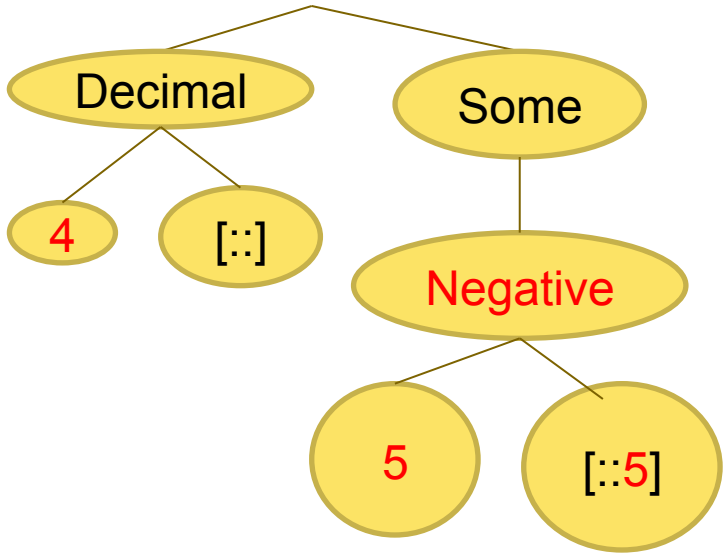
Un exemple

Eval vm_compute in get numeric_literal ".4E-55 + 3".

= Some

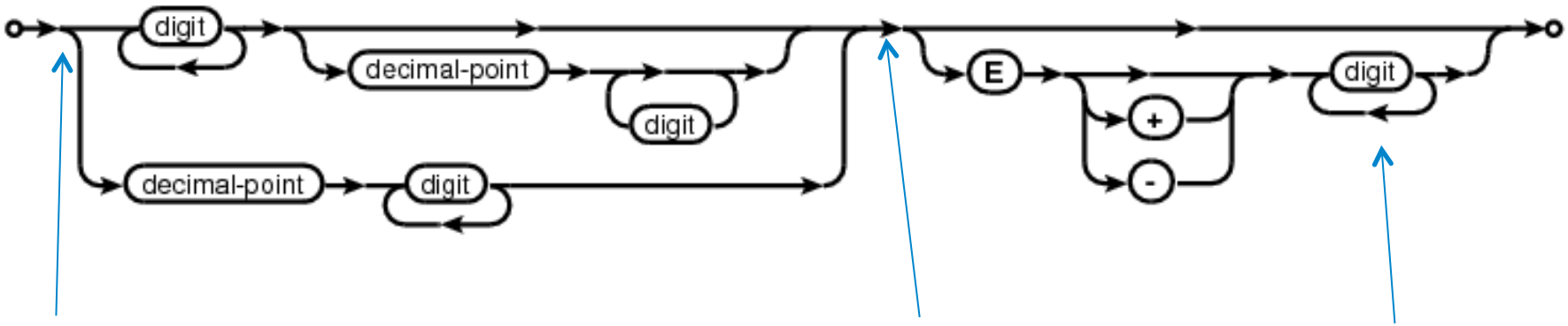
(Decimal (four, [::]),
Some (Some negative,
(five, [:: five])),
" + 3"%string)

: option (num_literal_data * string)



" + 3"%

Exemple tiré de la documentation de SQLite Litéral Numerique



Choix
<+>

Concaténation
<*>

répétition

On fournit des combinateurs en Coq pour écrire formellement cette grammaire:

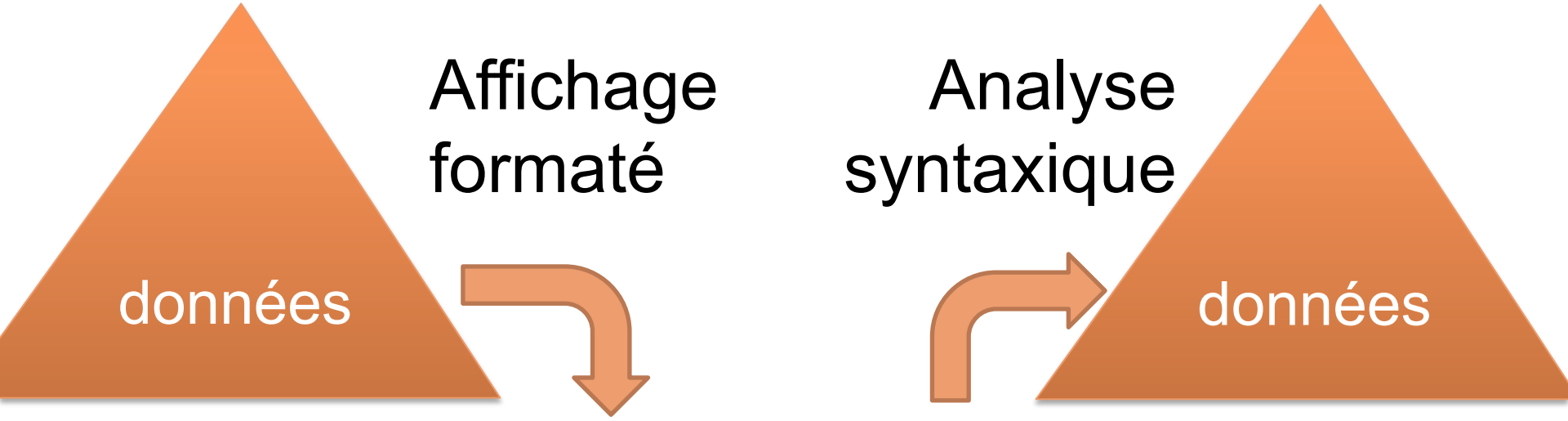
```
numeric_literal_ := (consequence (num_main <*> num_exponent) _ _)
        num_main := (consequence (lu <+> ld) _ _)
              lu := (consequence (numerics1 <*> decimals) _ _)
              ld := (consequence (dot *> numerics1) _ _)
        numerics1 := (consequence (easy_many1 numeric_char _ _))
```

↑

Juste des inférences logiques

_ Preuve fournié par l'utilisateur

L’affichage réversible



“chaîne de charactères”

Le type des affichages réversibles

```
Record syntax A precond :=

  { print : (A * string) -> string;

    parse : string -> option (A * string);

    reverse : forall a str,
                precond (a, str) ->
                parse (print (a, str)) = Some (a, str);

}.
```

Affichage formaté

Analyse syntaxique

Réversibilité conditionnelle

Le type des affichages réversibles

```
Record syntax A precond prop :=

  { print : (A * string) -> string;

    parse : string -> option (A * string);

    reverse : forall a str,
                precond (a, str) ->
                parse (print (a, str)) = Some (a, str);

    spec : forall str a rest,
                parse str = Some (a, rest) ->
                prop str (a, rest)
}.
```

Affichage formaté

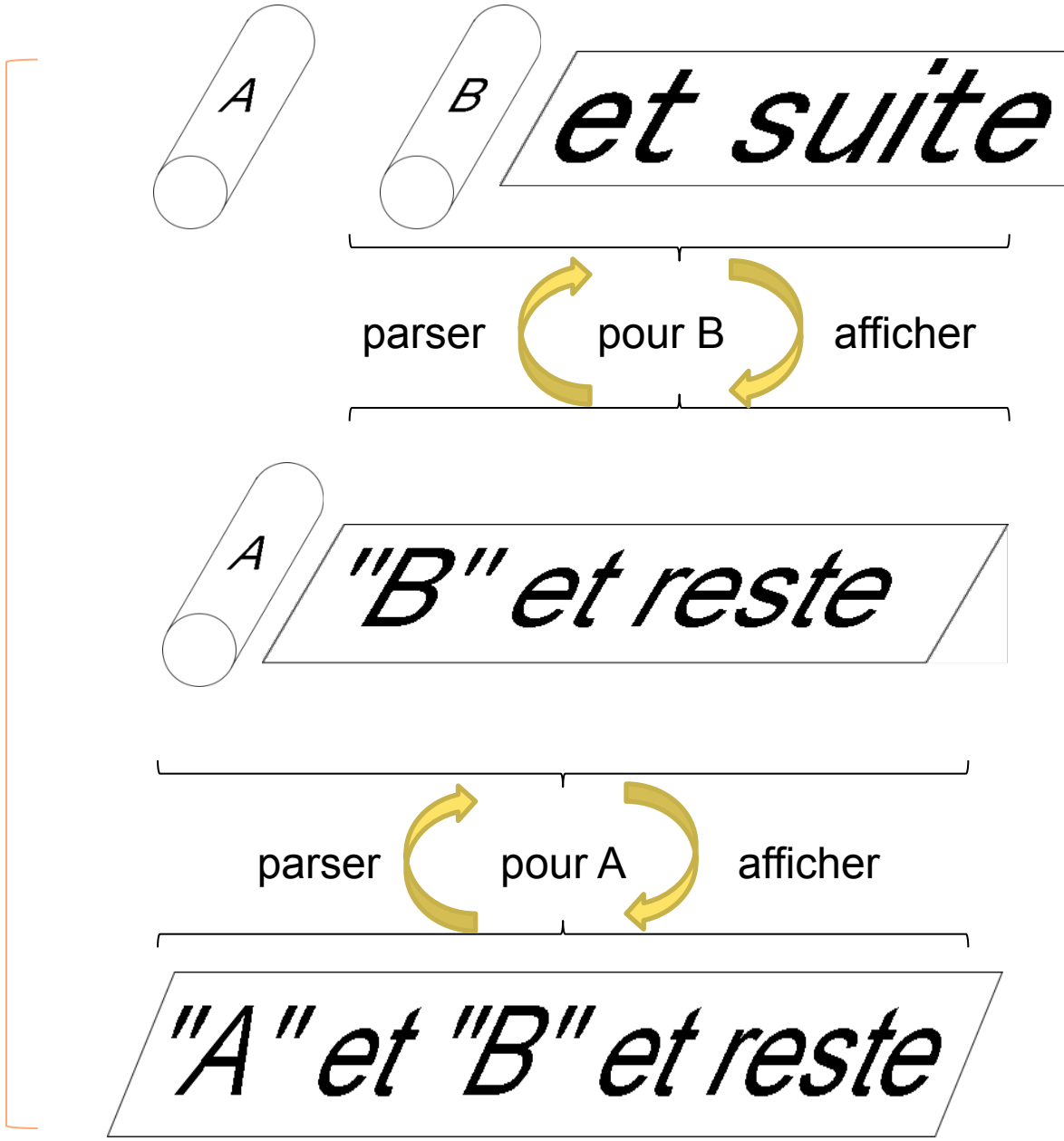
Analyse syntaxique

Réversibilité conditionnelle

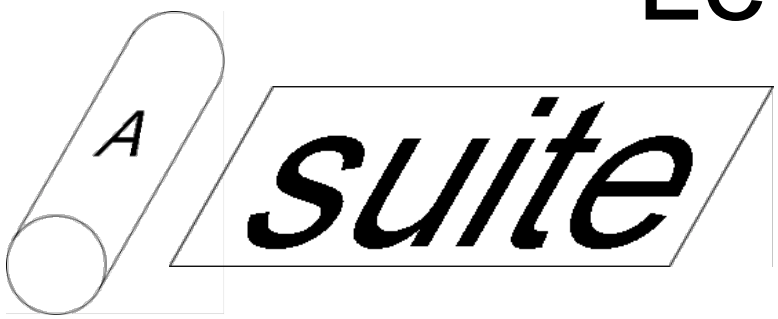
La propriéte “prop” est vraie pour “str” et “(a, rest)”

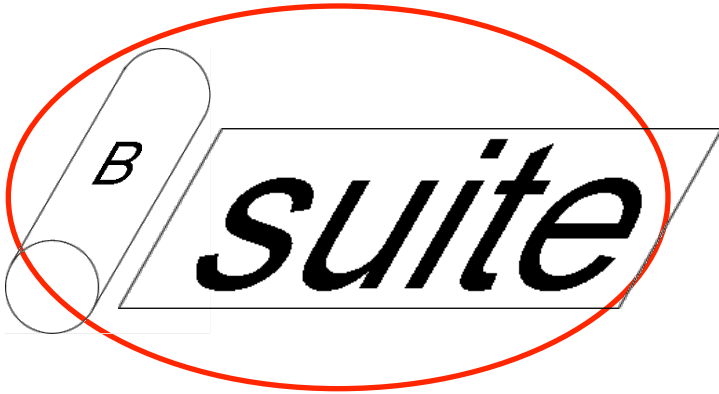
La concaténation

Affichage formaté et analyse syntaxique pour les paires $(A * B)$



Le choix





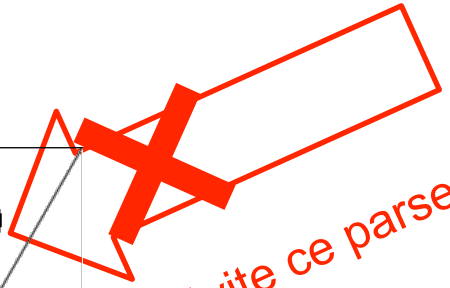
formater

Introduction d'une pré-condition

formater

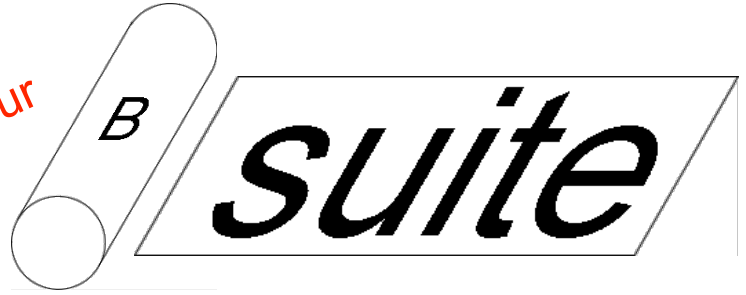
phrase pour A ou B, et suite

analyser



qui évite ce parseur

analyser



Affichage réversible:
Comparaison avec les travaux similaires

Nom	Année	Réversibilité	<*>	<+>	Répétition	Commentaire
Hirai	2014	Donné une condition explicite	Oui	Oui	Oui	JFLA 2014
Matsuda, Wang	2013	Si l'analyseur est sans ambiguïté	Oui	Oui	Oui	Pas de preuve formelle
Danielsson	2010, 2013	Si l'analyseur est sans ambiguïté	Oui	Oui	Oui	les grammaires infinies, prouvé en Agda
Affeldt, Nowak, Oiwa	2013	Sans conditions	Oui	Non	Oui	en Coq, “Somme dépendante”
Kennedy, Benton et al	2013	Sans conditions	Pas mentionées			Prouvé en Coq pour un langage spécial

« SELECT * FROM menu ; »

Cette présentation

Analyse syntaxique

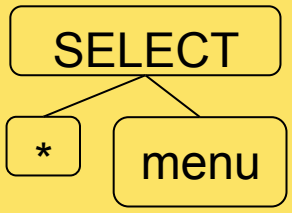
Normalisation

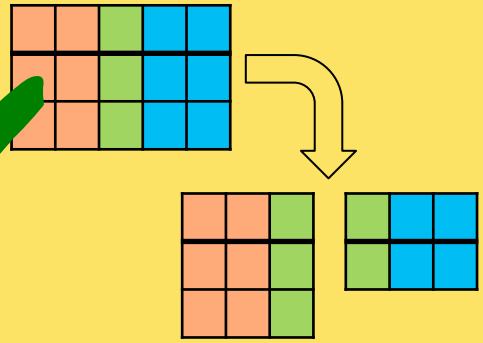
Optimisation des requêtes

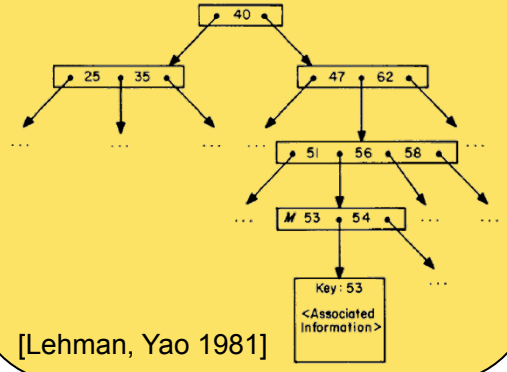
Arbre-indexe

Concurrence
La récupération d'un crash

Base de données







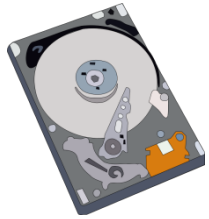
[Lehman, Yao 1981]

Fig. 2. An example B*-tree (with parameter $k = 2$).

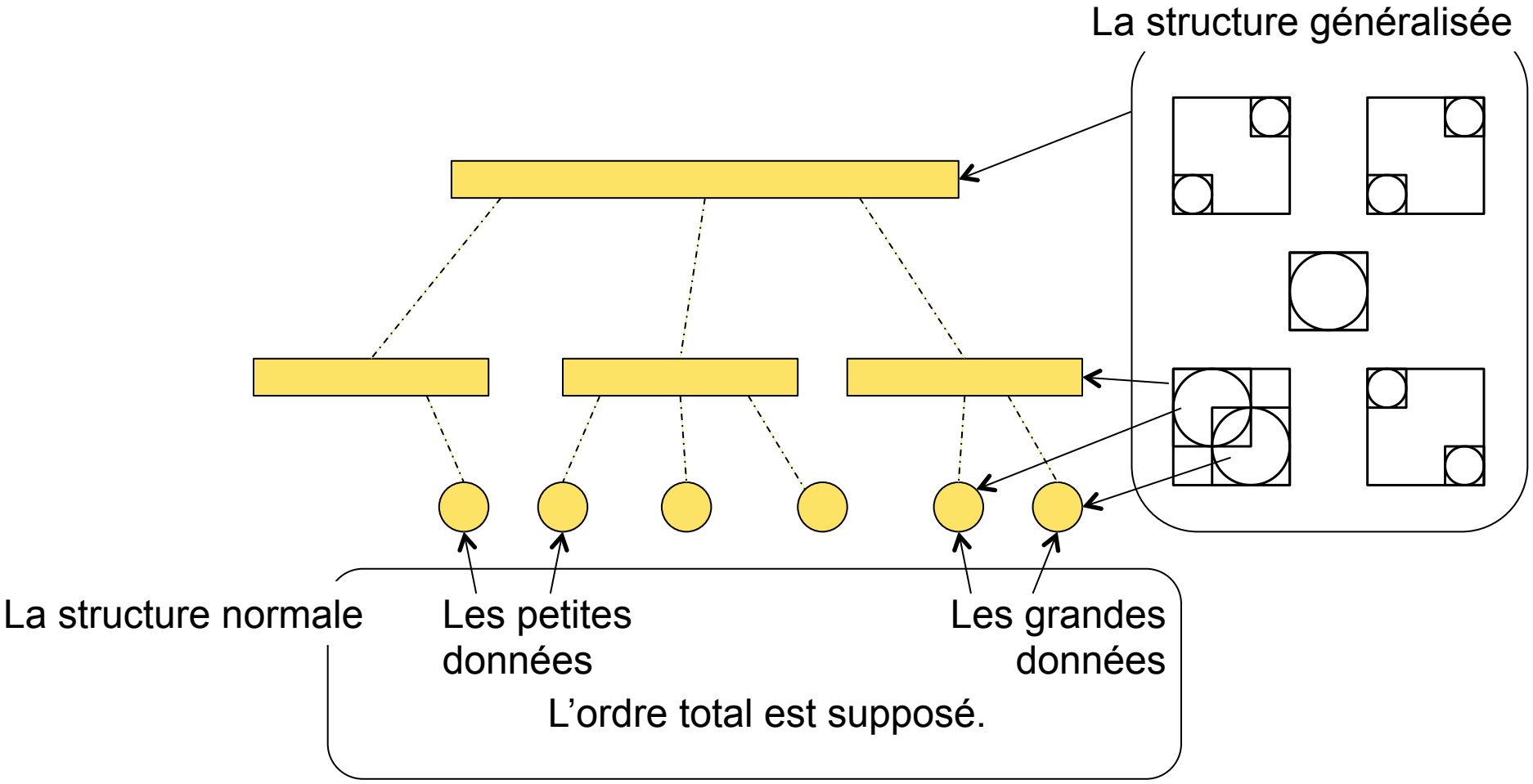
Couche du langage de manipulation

Couche du modèle relationnel

Couche des disques et de la mémoire



Les Structures d'Indexe Généralisées [Hellerstein, 1995]



Les clés et les données générales ont quelques axiomes

- **Consistent**(E,q): given an entry $E = (p, \texttt{ptr})$, and a query predicate q, returns false if $p \wedge q$ can be guaranteed unsatisfiable, and true otherwise. Note that an accurate test for satisfiability is not required here: Consistent may return true incorrectly without affecting the correctness of the tree algorithms. The penalty for such errors is in performance, since they may result in exploration of irrelevant subtrees during search.

- **Union**(P): given a set P of entries $(p_1, \texttt{ptr}_1)$, ... $(p_n, \texttt{ptr}_n)$, returns some predicate r that holds for all tuples stored below $\texttt{ptr}_1$ through $\texttt{ptr}_n$. This can be done by finding an r such that $(p_1 \vee \ldots \vee p_n) \rightarrow r$.

Ces axiomes sont facilement exprimés en Coq.

« SELECT * FROM menu ; »

Cette présentation

Analyse syntaxique

Normalisation

Optimisation des requêtes

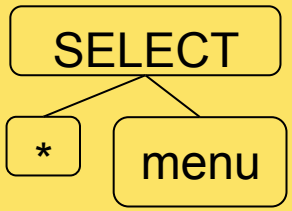
Arbre-indexe

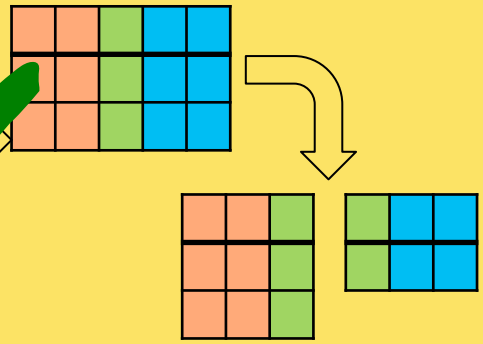
Modèle en Coq (avec ajout et recherche de données)

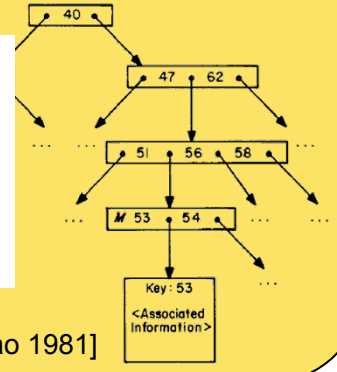
Concurrence ×

La récupération d'un crash ×

Base de données







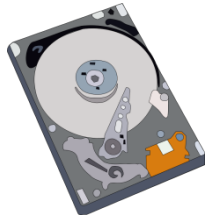
[Lehman, Yao 1981]

Fig. 2. An example B*-tree (with parameter $k = 2$).

Couche du langage de manipulation

Couche du modèle relationnel

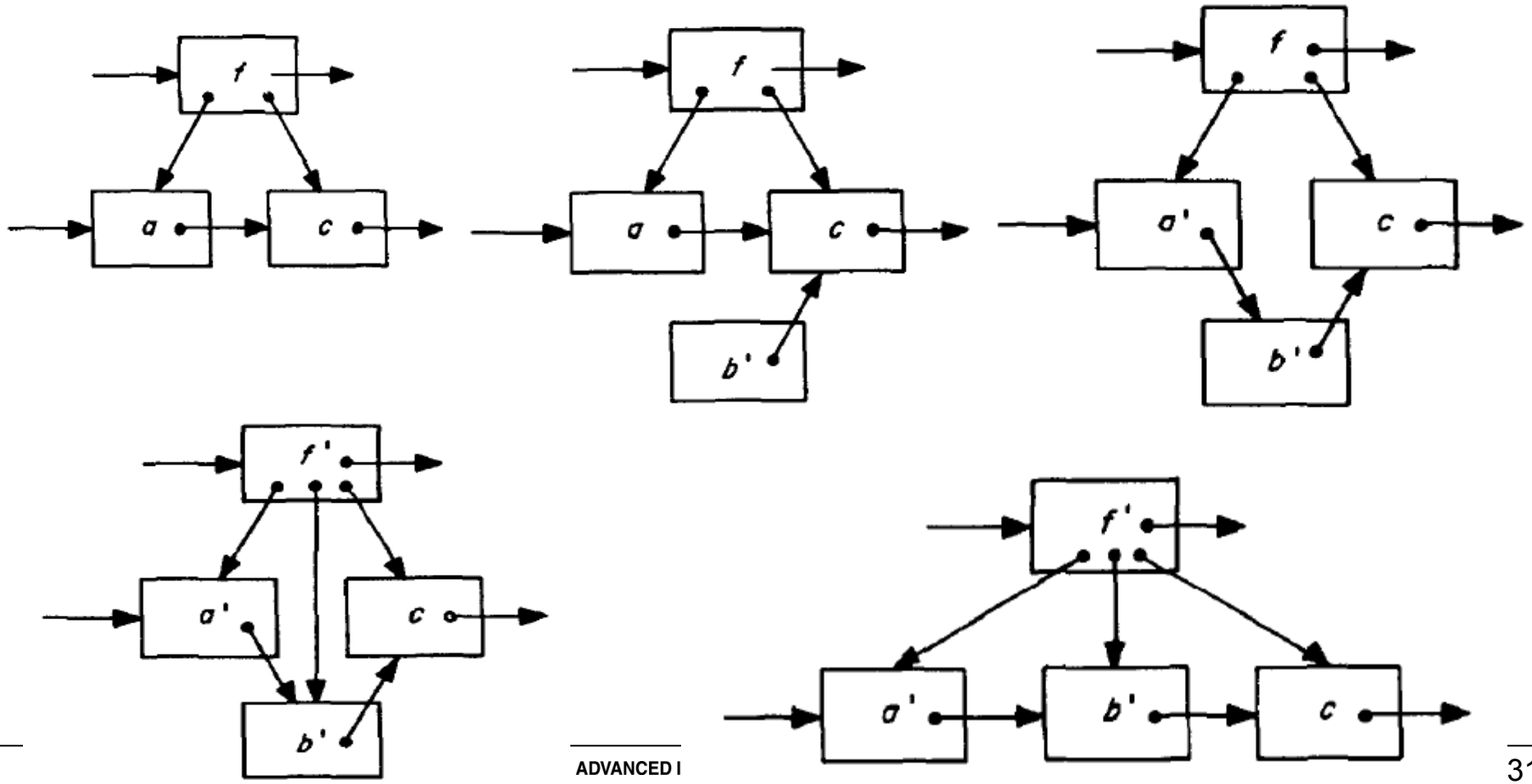
Couche des disques et de la mémoire



Permettre de lire pendant écrire

- Le fil d'écriture divise le node a dans a' et b'

[Lehman, Yao 1981]



La stratégie de formalisation

- Il faut oublier la sémantique relationnelle
- Les attributs sont juste les habitants d'un type (avec égalité décidable)
- Une dépendance fonctionnelle est une paire de séquences d'attributs

Le choix <|>

Printer-parser for A

syntax A P Q
print : (A * string) -> string
parse : string -> option (A * string)
reverse : print-parser is reversible on (a, str) satisfying P
spec : when parse succeeds, Q is satisfied

<|>

Printer-parser for B

syntax B P' Q'
print' : (B * string) -> string
parse' : string -> option (B * string)
reverse' : print'-parse' is reversible on (b, str) satisfying P'
spec' : when parse' succeeds, Q' is satisfied

=

Printer-parser for (A + B)

syntax (A + B) ?? ???
$print_{\dagger}$: ((A + B) * string) -> string
$parse_{\dagger}$: string -> option ((A + B) * string)
$reverse_{\dagger}$: $print_{\dagger}$-$parse_{\dagger}$ is reversible on (inl b, str) satisfying ?? // and for A
$spec_{\dagger}$: When $parse_{\dagger}$ succeds, Q or Q' is satisfied

b satisfies P' and $print_{\dagger}$-parse does not produce A result.

Améliorations Potentielles?

- Comment exécuter les analyseurs en C ou ruby?
- La grammaire deverait être un type de données (peut-être coinductive à la [Danielsson 2013]).

La concaténation 〈∗〉

Affiche formaté/analyse syntaxique pour A

```
syntax A P Q
print     : (A * string) -> string
parse     : string -> option (A * string)
reverse   : (parse o print) est réversible
            sur (a, str) satisfaisant P
spec      : Quand parse réussit,
            Q est satisfait
```

〈∗〉

Affiche formaté/analyse syntaxique pour B

```
syntax B P' Q'
print'    : (B * string) -> string
parse'    : string -> option (B * string)
reverse'  : print'-parser' is reversible on
            (b, str) satisfying P'
spec'     : When parse' succeeds, Q' is
            satisfied
```

Affiche formaté/analyse syntaxique pour A*B

=

```
syntax (A * B) ?1  ?2
print†    : (A * B * string) -> string
parse†    : string -> option (A * B *
            string)
reverse†: print† -parse† is reversible
            on (a, b, str) satisfying ?1
spec†     : When parse† succeeds, ?2.
            Is satisfied
```

If (b, str) satisfies P' and Q' mid_str (b, str) then (a, mid_str) satisfies P

The composition of Q and Q' (as relations).

La troisième forme normale

Catalogue des formes normales:

1. 1FN : la première forme normale
2. 2FN : la deuxième forme normale
3. 3FN : la troisième forme normale
4. FNBC : la forme normale de Boyce Codd
5. 4FN : la quatrième forme normale
6. 5FN : la cinquième forme normale
7. FNDC: la forme normale domaine clé
8. 6FN : la sixième forme normale (rarement présentée)

Référence:
http://fr.wikipedia.org/wiki/Forme_normale_(bases_de_données_relationnelles)

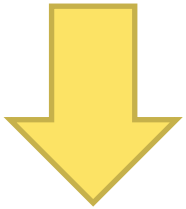
Les propriétés ACID pour les systèmes de base de données

- Atomicité
  Les changements sont appliqués complètement ou pas du tout. Les changements partiels doivent etre corrigés.
- Cohérence
  Les changements d'états valides résultent en des états valides.
- Isolation
  Même les changements concurrents simulent une exécution temporellement en série.
- Durabilité
  Les changements opérés sont permanents, sauf en cas de nouveaux changements.

Les *anomalies* : Problèmes de cohérence

- Une anomalie d'effacement

profID	Nom de l'enseignant	En poste depuis	Cours	Jour	Horaire
33	R. Wavey	1951-09-01	Physique 2A	Mercredi	15:00-
34	…	…	…	…	…



On enlève juste un cours, mais un enseignant disparaît

profID	Nom de l'enseignant	En poste depuis	Cours	Jour	Horaire
34	…	…	…	…	…

Dépendance fonctionnelle

profID	Nom de l'enseignant	En poste depuis	Cours	Jour	Horaire
33	R. Wavey	1951-09-01	Physique 2A	Mercredi	15:00-
34	…	…	…	…	…

{profID} →
{Nom de l'enseignant, En poste depuis}
{profID, Jour, Horaire} →
{Cours}
{profID, Jour, Horaire} →
{profID, Nom de l'enseignant, En poste depuis, Cours, Jour, Horaire}

Premier Algorithme de Bernstein [Bernstein, 1976]

1. (Eliminate extraneous attributes.) Let F be the given set of FDs. Eliminate extraneous attributes from the left side of each FD in F, producing the set G. An attribute is extraneous if its elimination does not alter the closure of the set of FDs.

2. (Find covering.) Find a nonredundant covering H of G.

3. (Partition.) Partition H into groups such that all of the FDs in each group have identical left sides.

4. (Construct relations.) For each group, construct a relation consisting of all the attributes appearing in that group. Each set of attributes that appears on the left side of any FD in the group is a key of the relation. (Step 1 guarantees that no such set contains any extra attributes.) All keys found by this algorithm will be called synthesized. The set of constructed relations constitutes a schema for the given set of FDs.

Coq lines

Les Propriétés Préservées

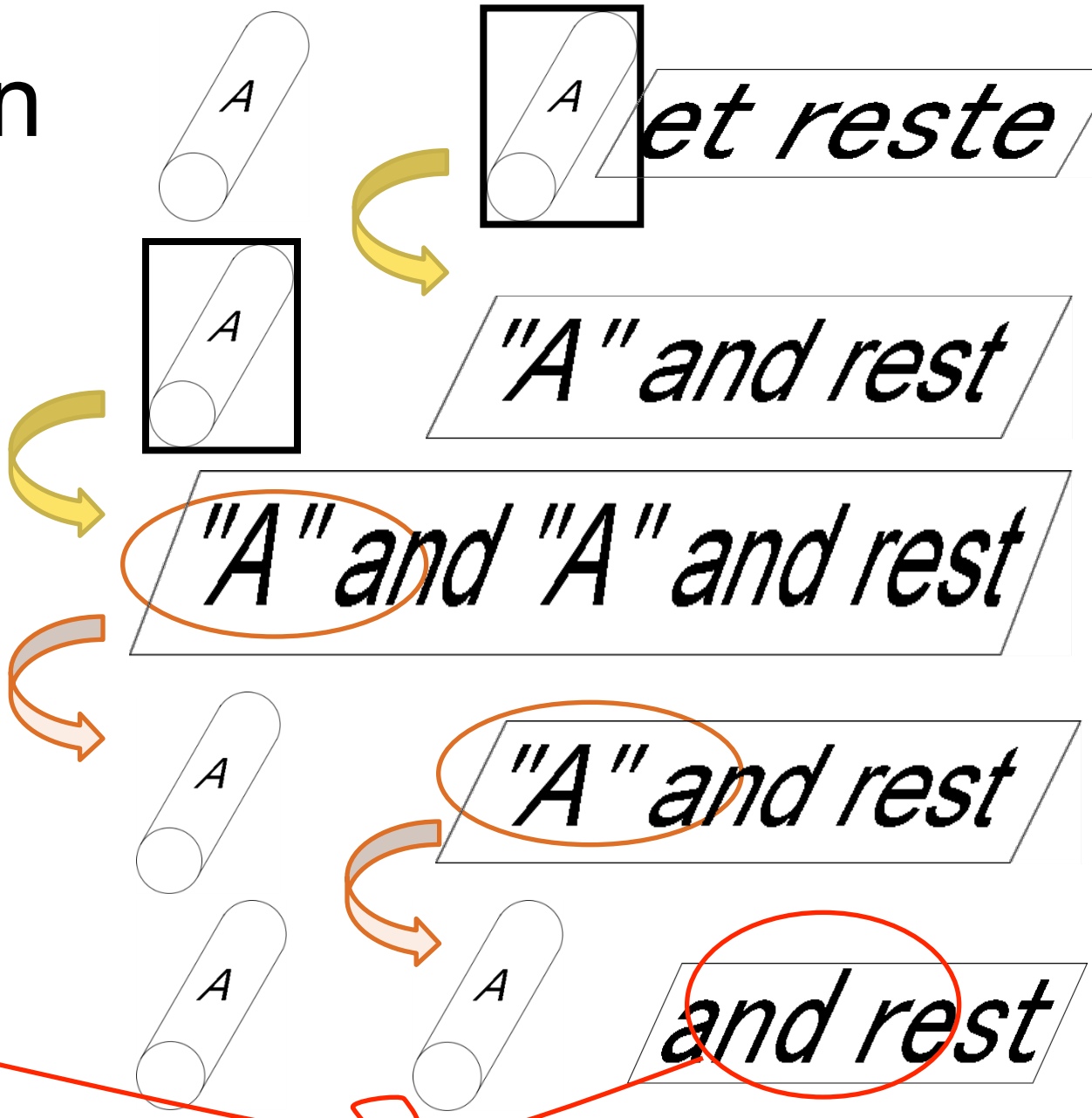
- Chaque étage préserve la clôture des dépendances fonctionnelles!
- Trés facile à formaliser et prouver. (めんどうだけど)

Perspective: Second algorithm de Bernstein

- Le nombre des relations produites par le premier algorithme de Bernstein n'est pas optimal
- Le second algorithme de Bernstein donne un nombre de relation optimal (= le plus petit) (réponse au challenge de Codd).
- Nous avons formalisé le second algorithmique mais pas encore sa preuve.

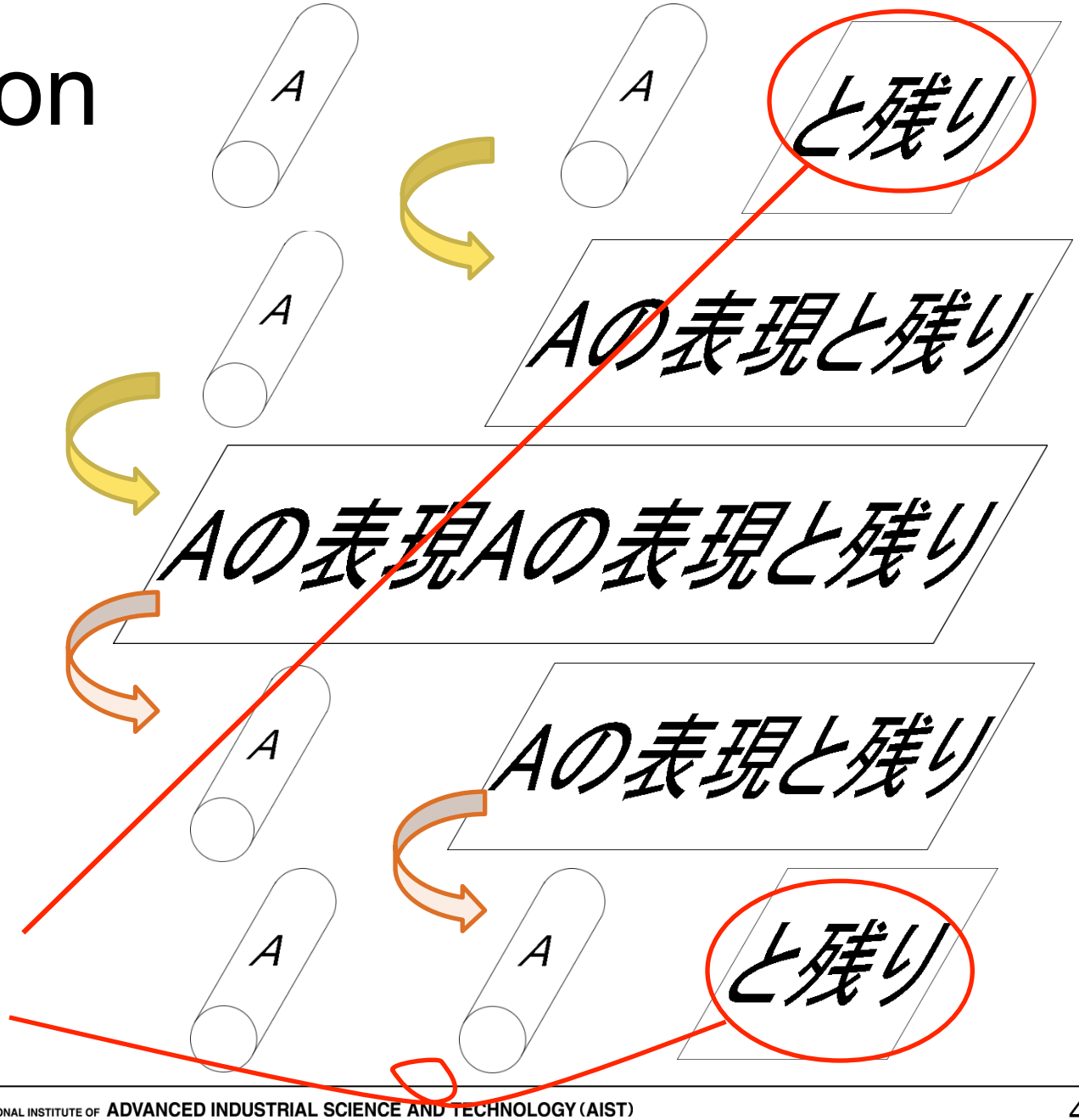
- Les multi-dépendances, formes normales 4 et 5.

La répétition



Finish because A is not readable

La répétition



A is not readable

Pourquoi ces objets graphiques?

"S'il y a une dérivation (graphique) qui utilise une dépendance g," [Bernstein, trans.]

Une reformulation:

"Si toutes les dérivations (graphiques) utilisent une dépendance g,"