Une syntaxe améliorée pour la description de circuits digitaux

Aliaume Lopez
Sous la direction du professeur Dan R. Ghica

University of Birmingham, Computer Science, UK

1 Juin 2016 – 16 Juillet 2016

Table des matières

L'Université de Birmingham

Le campus de l'université

L'équipe de travail

Dan R. Ghica

- 3 autres étudiants
- Achim Jung
- Le « Theory Group »

Motivation

Étude syntaxique des circuits digitaux (avec un sens)

- Fournir une syntaxe
- Fournir une sémantique opérationnelle

Quels outils ?

- Théorie des catégories
- Système d'écriture de graphes (efficacité)

Théorie des catégories et diagrammes

Catégorie

- Objets
- Morphismes

Diagramme

Constat

Pour avoir une correspondance entre un *type* de diagramme et un *type* de catégorie il faut bien choisir les axiomes [1].

Théorie des catégories et diagrammes

Catégorie

- Objets
- Morphismes

Diagramme

- Fils / Traits
- Nœuds / Boîtes

Constat

Pour avoir une correspondance entre un *type* de diagramme et un *type* de catégorie il faut bien choisir les axiomes [1].

Théorie des catégories et diagrammes

Catégorie

- Objets
- Morphismes
- Opérateurs (composition ...)

Diagramme

- Fils / Traits
- Nœuds / Boîtes
- Agencement spatial

Constat

Pour avoir une correspondance entre un *type* de diagramme et un *type* de catégorie il faut bien choisir les axiomes [1].

Théorie des catégories et diagrammes

Catégorie

- Objets
- Morphismes
- Opérateurs (composition ...)
- Transformation naturelles

Diagramme

- Fils / Traits
- Nœuds / Boîtes
- Agencement spatial
- Transformation de circuits

Constat

Pour avoir une correspondance entre un *type* de diagramme et un *type* de catégorie il faut bien choisir les axiomes [1].

Théorie des catégories et diagrammes

Catégorie

- Objets
- Morphismes
- Opérateurs (composition ...)
- Transformation naturelles
- Axiomes d'équivalence

Diagramme

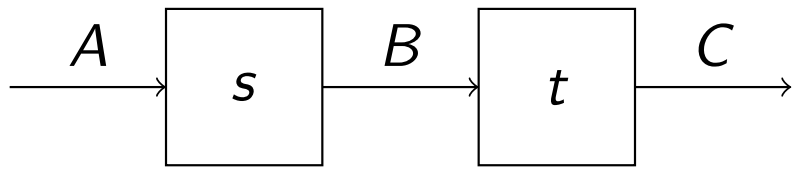
- Fils / Traits
- Nœuds / Boîtes
- Agencement spatial
- Transformation de circuits
- Déformations autorisées

Constat

Pour avoir une correspondance entre un *type* de diagramme et un *type* de catégorie il faut bien choisir les axiomes [1].

Théorie des catégories et diagrammes

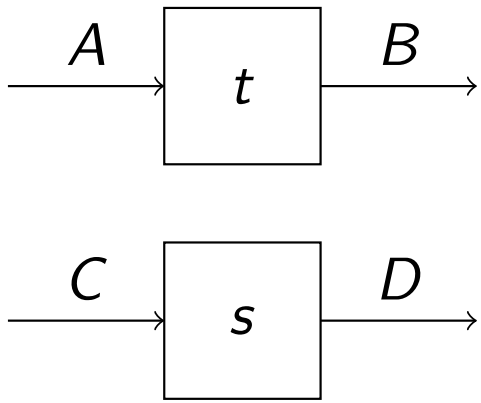
$$t \circ s : A \to C$$



$$s : A \to B \quad t : B \to C$$

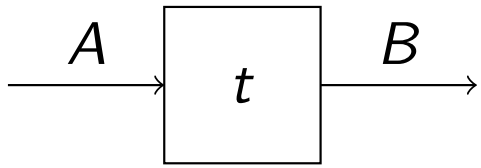
Composition séquentielle

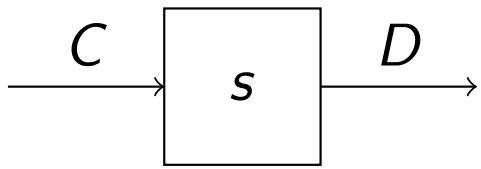
Théorie des catégories et diagrammes



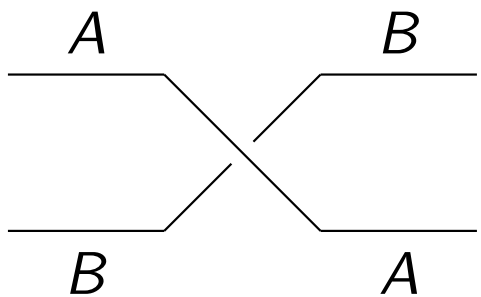
$t \otimes s$

Théorie des catégories et diagrammes



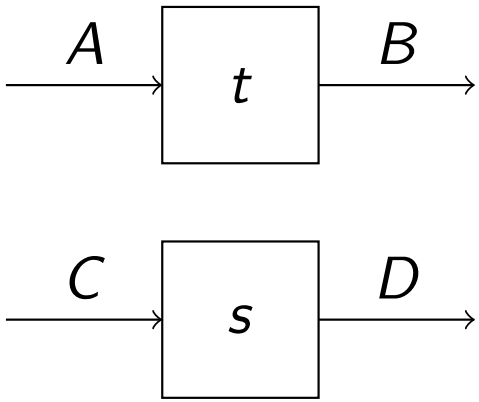


$t \otimes s$

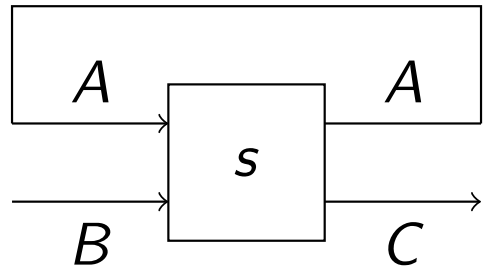


$s_{A,B}$

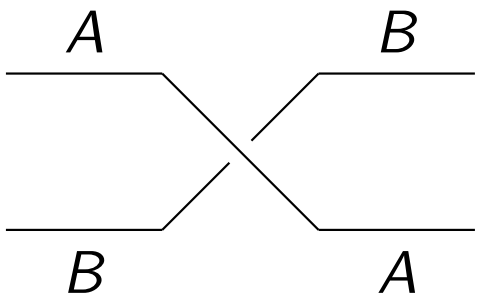
Théorie des catégories et diagrammes



$t \otimes s$

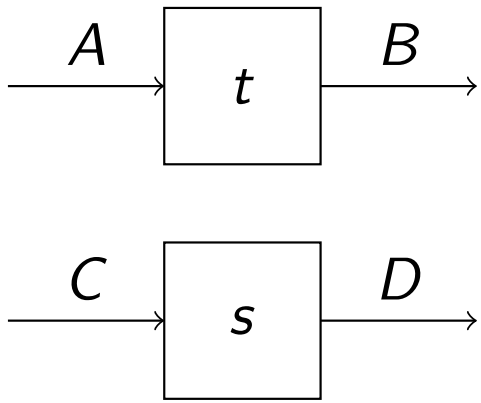


$\mathrm{Tr}_A(s)$

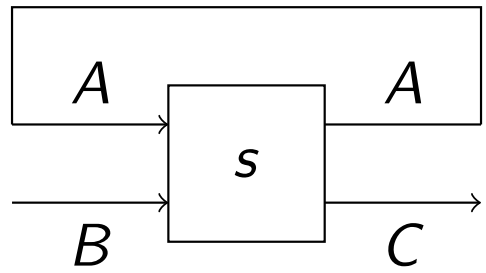


$s_{A,B}$

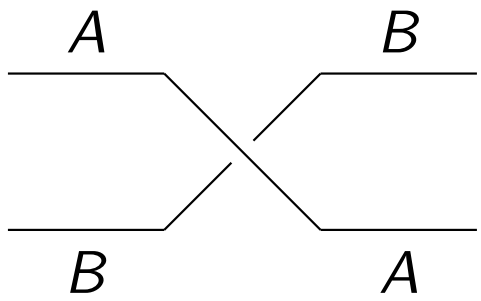
Théorie des catégories et diagrammes



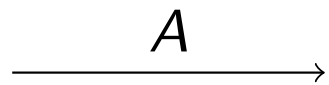
$t \otimes s$



$\mathrm{Tr}_A(s)$



$s_{A,B}$



Id_A

Soumission au POPL 17 (1).

Signature de la catégorie

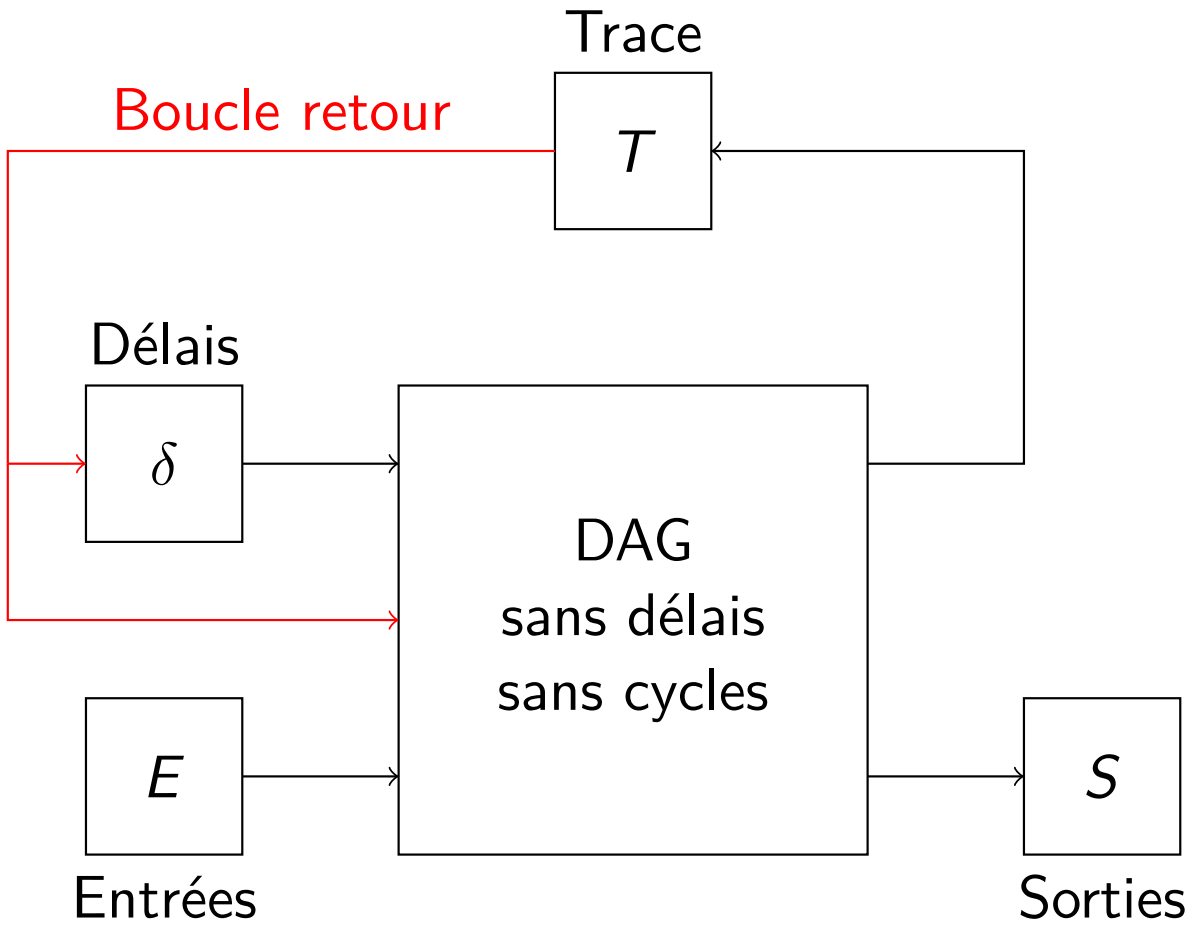
- Objets : $\mathbb{N}$
- Morphismes : valeurs, fork, join, $s_{A,B}$ ...
- Opérateurs : $\cdot$, $\otimes$
- Transformation naturelles : Tr
- Axiomes d'équivalence : monotonicité, extentionnalité ...

Soumission au POPL 17 (2)

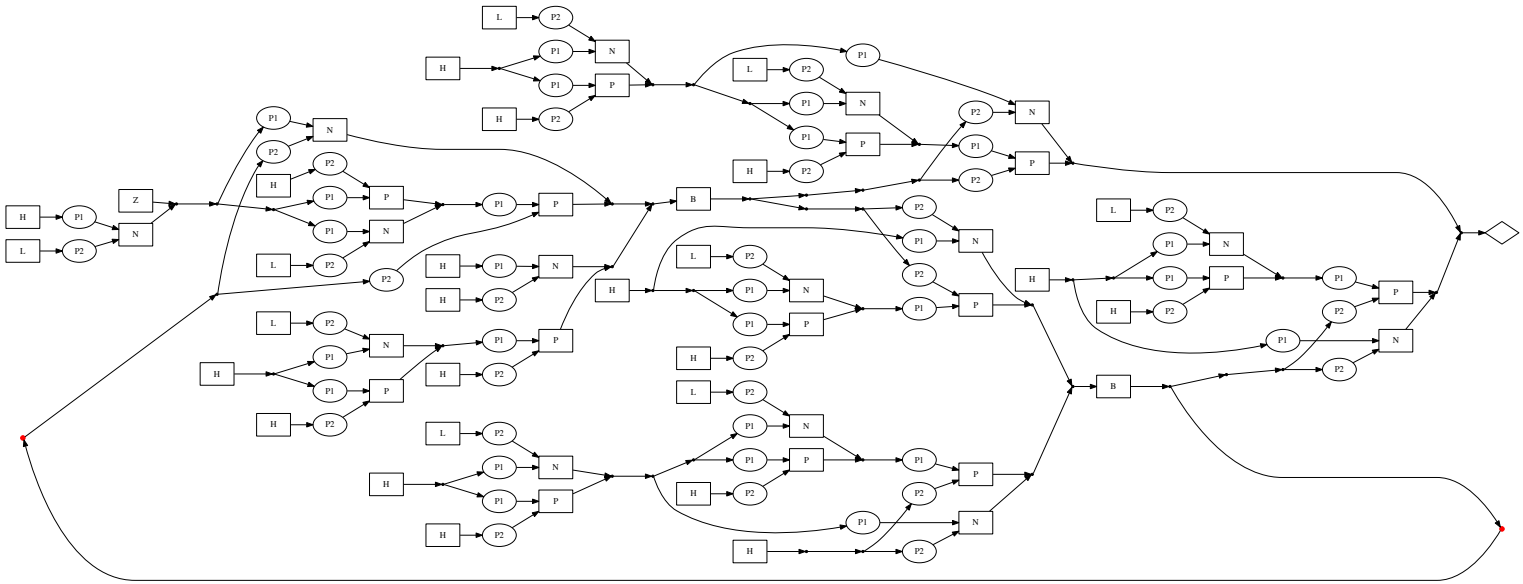
Morphismes de base

- Treillis de valeurs V : High, Low, $\bot$, $\top$
- Fork $f : 1 \to 2$
- Join $j : 2 \to 1$
- Oubli $w : 1 \to 0$
- Délai $\delta : 1 \to 1$
- Circuits de type « boîte noire » (multiplexer . . .)

Soumission au POPL 17 (3)

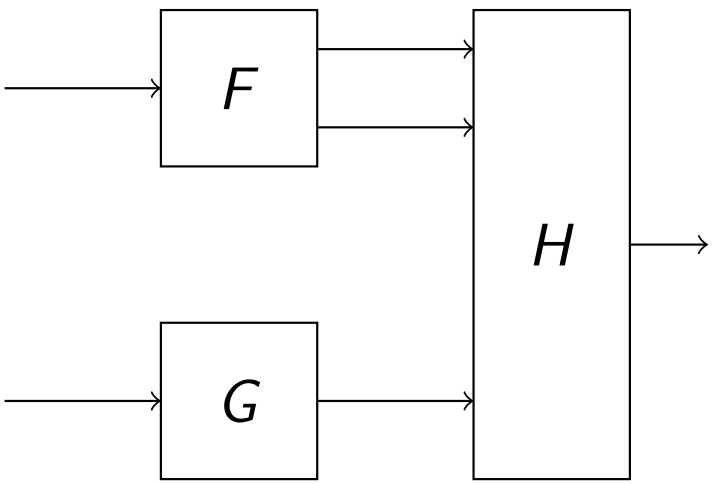


Soumission au POPL 17 (4)



Un problème de syntaxe

Circuit

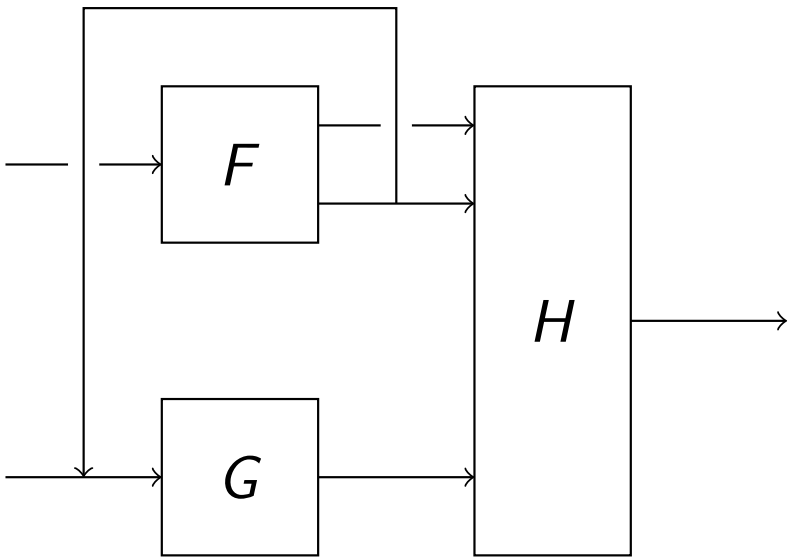


Expression associée

$$(F \otimes G) \cdot H$$

Un problème de syntaxe

Circuit



Expression associée

$$
\begin{aligned}
&\mathrm{Tr}_1\big[\\
&\quad (s_{1,1} \otimes \mathrm{Id}_1)\cdot \\
&\quad (\mathrm{Id}_1 \otimes j)\cdot \\
&\quad (F \otimes G)\cdot \\
&\quad (\mathrm{Id}_1 \otimes f \otimes \mathrm{Id}_1)\cdot \\
&\quad (s_{1,1} \otimes \mathrm{Id}_2) \\
&\quad \big] \cdot H
\end{aligned}
$$

La syntaxe proposée

Construction *point-free* d'un morphisme

$$F = \mathrm{Tr}^k \left[\bigodot_i \bigotimes_j g_{i,j} \right]$$

La syntaxe proposée

Construction *point-free* d'un morphisme

$$F = \mathrm{Tr}^k \left[\bigodot_i \bigotimes_j g_{i,j} \right]$$

Construction *point-wise* d'un morphisme

$$F = \lambda x.G$$

La syntaxe proposée

Construction *point-free* d'un morphisme

$$F = \mathrm{Tr}^k \left[\bigodot_i \bigotimes_j g_{i,j} \right]$$

Construction *point-wise* d'un morphisme

$$F = \lambda x.G$$

Symétrie des circuits dirigés

Les circuits ont un sens, mais une symétrie par rapport à un axe vertical permet de construire un circuit valide.

La syntaxe proposée

Concepts

- Variables productrices

Syntaxe

La syntaxe proposée

Concepts

- Variables productrices

Syntaxe

- `:a`

La syntaxe proposée

Concepts

- Variables productrices
- Variables consommatrices

Syntaxe

- `:a`

La syntaxe proposée

Concepts

- Variables productrices
- Variables consommatrices

Syntaxe

- `:a`
- `a:`

La syntaxe proposée

Concepts

- Variables productrices
- Variables consommatrices
- Lien entre plusieurs paires de variables

Syntaxe

- `:a`
- `a:`

La syntaxe proposée

Concepts

- Variables productrices
- Variables consommatrices
- Lien entre plusieurs paires de variables

Syntaxe

- `:a`
- `a:`
- `link a:b for F`

La syntaxe proposée

Concepts

- Variables productrices
- Variables consommatrices
- Lien entre plusieurs paires de variables

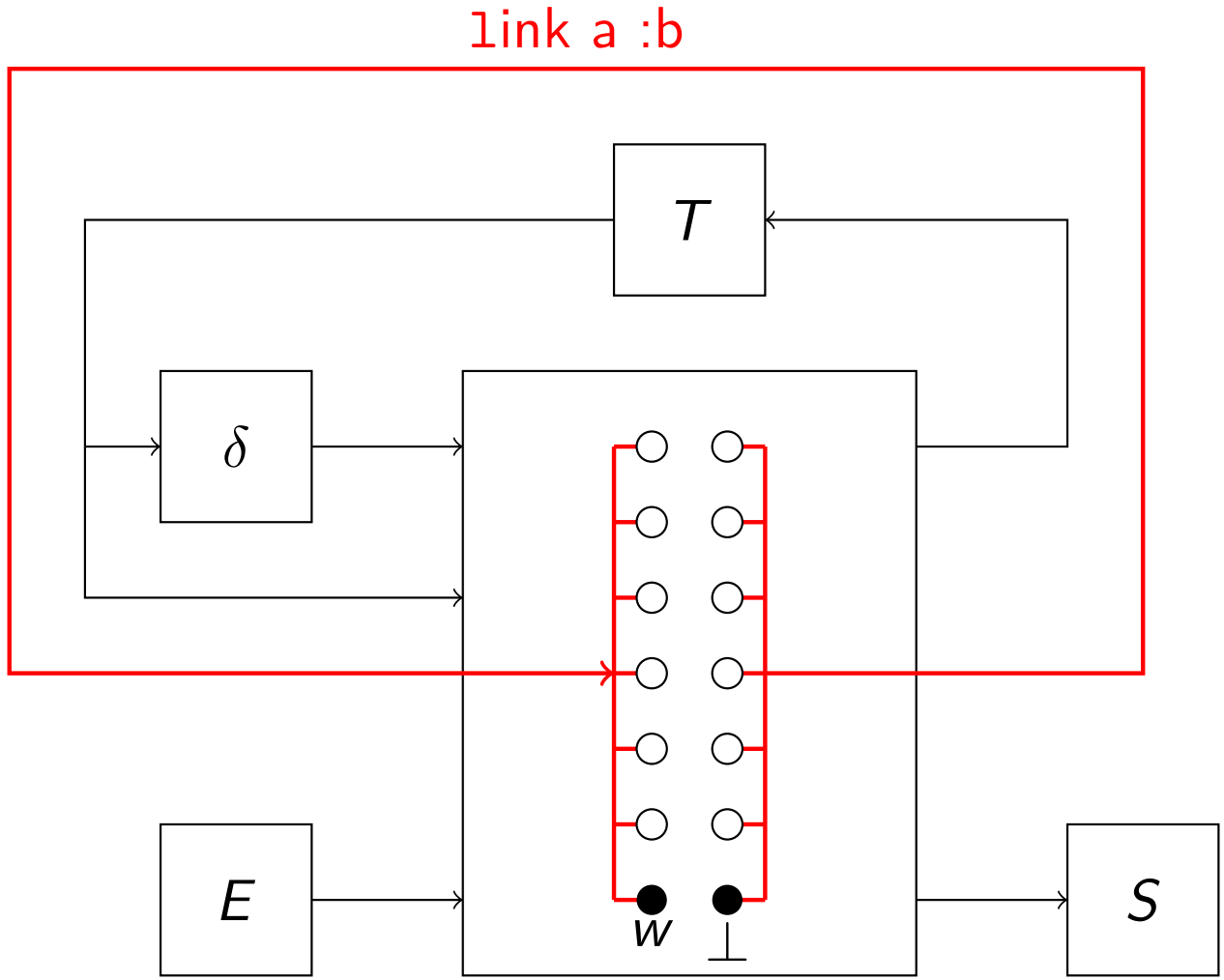
Syntaxe

- `:a`
- `a:`
- `link a:b for F`

Explication

Cette description apporte la possibilité de manipuler des *points* dans la construction des morphismes. Où un point correspond à un *fil* du circuit.

La syntaxe proposée



La syntaxe proposée

Un système de type

$$\frac{}{\Gamma, a \mid \Delta \vdash a : 0 \to 1}$$

$$\frac{}{\Gamma \mid \Delta, a \vdash a : 1 \to 0}$$

$$\frac{\Gamma, b_1 \dots b_n \mid \Delta, a_1 \dots a_n \vdash \phi : n \to m}{\Gamma \mid \Delta \vdash \text{link } a_1 : b_1 \ \dots \ a_n : b_n \ \phi : n \to m}$$

Ne pas oublier

- Composition séquentielle
- Composition parallèle
- Identité (poly)
- Symétrie
- Morphismes de base

Modifications apportées au programme

Les différentes étapes de modification

1. Intégration de la nouvelle syntaxe
   - 1.1 Lexer et parser
   - 1.2 Système de type
   - 1.3 Transformation en graphe
2. Correction du code
3. Nouvelle structure de données
4. Changement de style d'écriture

Modifications apportées au programme

Fonctionnement de la ré-écriture

1. Modifications locales
   - 1.1 Reconnaissance de motif
   - 1.2 Ré-écriture
2. Modifications globales

Problème de performance

Chaque étape demande de considérer les voisins au sens large d'un nœud, un moyen efficace d'ajouter, modifier et supprimer des arêtes et des nœuds.

Modifications apportées au programme

Exemple de ré-écriture des identités

```
let remove_identity ~node:n t =
    try (* pattern matching failure means no modification *)
        let [pre] = edges_towards  ~node:n t in
        let [pos] = edges_from     ~node:n t in
        let None  = id_find n t.labels       in
        if List.mem n t.nodes then
            t |> edge_remove_node ~first:pre ~using:n ~second:pos
              |> main_rem ~node:n
        else
            t
    with
        Match_failure _ -> t;;
```

Conclusion

Résultats

- La syntaxe répond au problème et pose des questions intéressantes
- Correction du programme de démonstration

Ouvertures

- Structure de données persistante (complexité générale)
- Gestion des délais
- Bus de valeurs

Remerciements

- Ghica Dan pour m'avoir proposé un stage aussi intéressant
- Mayaux Damien pour sa patience et son aide au debug
- Millan Mégane pour son soutien moral
- Le « Theory Group » pour l'aide précieuse sur la théorie des catégories

Bibliographie


P. Selinger.
A survey of graphical languages for monoidal categories.
ArXiv e-prints, August 2009.