Présentation du compilateur Faust

Yann Orlarey, Stéphane Letz

1 avril 2021

Les grandes étapes

- Représentations internes basée sur des arbres non mutables et le hash-consing
- Parsing Lex/Yacc
- Evaluation du programme sous la forme d'un circuit de processeurs de signaux
- Propagation symbolique de signaux dans le circuit
- Normalisation et optimisation des signaux
- Typage et calcul d'intervals
- Traduction des signaux en code impératif (FIR)
- Génération du code par le backend choisi

Arbres

Arbres non-mutables, hash-consing, DAG, propriétés mutables.

Propriété des arbres : $t_1 = t_2 \Leftrightarrow M(t_1) = M(t_2)$

- symbols : tlib/symbol.hh+cpp
- nodes : tlib/node.hh+cpp
- trees : tlib/tree.hh+cpp, 2 types de recursivité (de Bruijn+ symbolique)
- constructeurs
- destructurateurs
- propriétés

Arbres : exemple de la composition séquentielle `A:B`

```
gGlobal->BOXSEQ = symbol("BoxSeq");

Tree boxSeq(Tree x, Tree y)
{
    return tree(gGlobal->BOXSEQ, x, y);
}
bool isBoxSeq(Tree t, Tree& x, Tree& y)
{
    return isTree(t, gGlobal->BOXSEQ, x, y);
}
```

Arbres : récursivité *de Bruijn*

```
Tree rec(Tree body)
{    return tree(gGlobal->DEBRUIJN, body); }

bool isRec(Tree t, Tree& body)
{   return isTree(t, gGlobal->DEBRUIJN, body); }

Tree ref(int level)
{   return tree(gGlobal->DEBRUIJNREF, tree(level)); }

bool isRef(Tree t, int& level)
{
    Tree u;
    if (isTree(t, gGlobal->DEBRUIJNREF, u)) {
        return isInt(u->node(), &level);
    } else {
        return false;
    }
}
```

Arbres : récursivité *symbolique*

```
Tree rec(Tree var, Tree body) {
    Tree t = tree(gGlobal->SYMREC, var);
    t->setProperty(gGlobal->RECDEF, body);
    return t; }

bool isRec(Tree t, Tree& var, Tree& body) {
    if (isTree(t, gGlobal->SYMREC, var)) {
        body = t->getProperty(gGlobal->RECDEF);
        return true;
    } else {
        return false; } }

Tree ref(Tree id) { return tree(gGlobal->SYMREC, id); }

bool isRef(Tree t, Tree& v) {
    return isTree(t, gGlobal->SYMREC, v); }
```

Parsing Lex/Yacc

- `parser/faustlexer.l`
- `parser/faustparser.y`
- `libcode.cpp/parseSourceFiles()`
- `parser/sourcereader.hh/SourceReader`
- environnement
- Chargeur récursif, utilisation des url

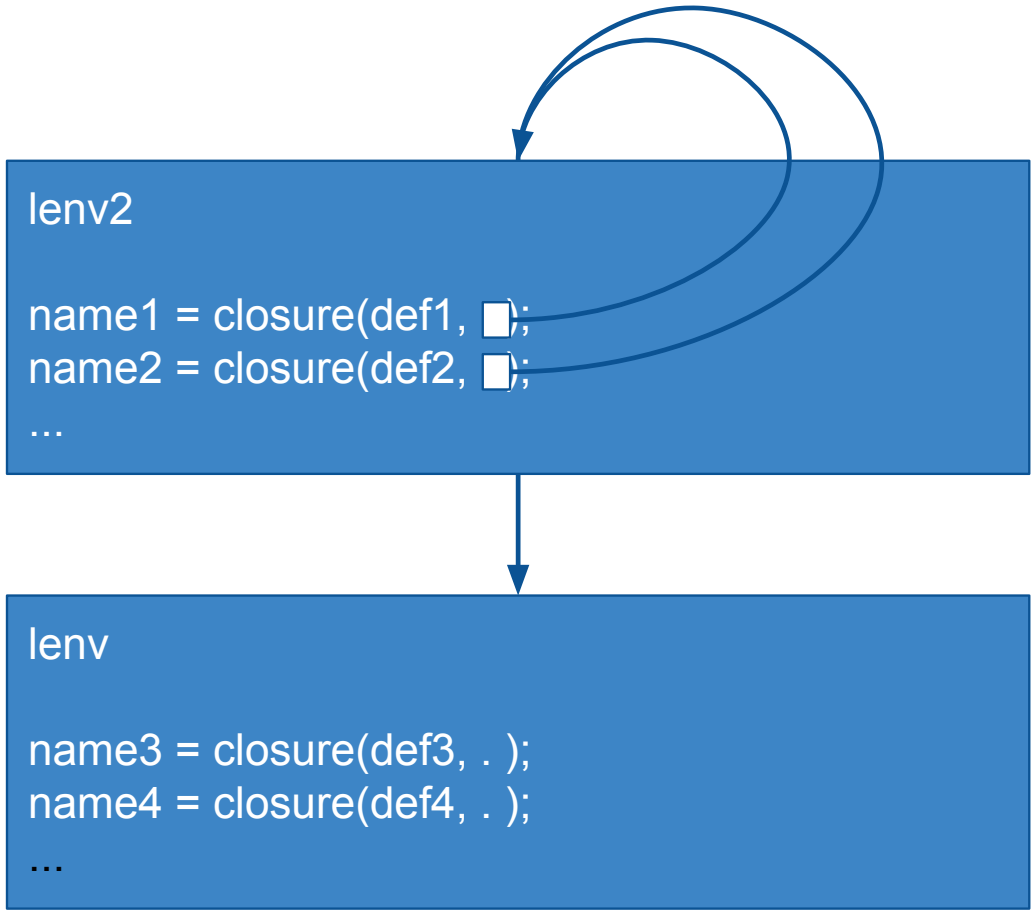
Parsing

Parsing de l'expression `library("math.lib")` :

```
(G := gGlobal)
const char* fname = tree2str(label);
Tree eqlst = G->gReader.expandList(G->gReader.getList(fname));
Tree res   = closure(boxEnvironment(), G->nil, G->nil,
                     pushMultiClosureDefs(eqlst, G->nil, G->nil));
setDefNameProperty(res, label);
return res;
```

Environnements

Les définitions d'un programme sont organisées en environnements par `pushMultiClosureDefs()` :



Evaluation

Evaluation de la définition de `process` dans l'environnement résultant de la lecture des fichiers sources (voir `eval.cpp`) :

```
Tree evalprocess(Tree eqlist)
{
    Tree b=a2sb(eval(boxIdent(G->gProcessName.c_str()), G->nil,
                pushMultiClosureDefs(eqlist, G->nil, G->nil)));

    if (G->gSimplifyDiagrams) {
        b = boxSimplification(b);
    }

    return b;
}
```

Propagation symbolique

bla bla