Comptes rendus

Camille VEDANI[1]

1 **Université III Paul Sabatier, France,** `camille.vedani@univ-tlse3.fr`

Résumé

Ce document correspond à la documentation d'`otawa` en français et détaillée.

1 Présentation générale

Ce document présente les principales classes du module `otawa-xdd`, une extension de OTAWA permettant de manipuler des structures appelées *XDD* (eXtended Decision Diagrams). Il s'agit de structures en graphe acyclique orienté (DAG) utilisées notamment pour représenter des ensembles d'événements ou des structures symboliques de manière compacte.

2 Représentation bas-niveau des XDD

Les classes suivantes sont utilisées pour la représentation interne des XDD :

- `otawa::xengine::Node` : représente un nœud du DAG. Chaque nœud encode une variable et deux successeurs (cas vrai / faux).
- `otawa::xengine::NodeManager` : gère l'allocation et la mémorisation des nœuds. Elle est essentielle pour assurer l'unicité et la réutilisation des nœuds identiques.
- `otawa::xengine::OpsManager` : assure la mémorisation des opérations entre nœuds de XDD (comme les additions ou intersections).

3 Classes d'encapsulation

Pour faciliter la manipulation des XDD, plusieurs classes fournissent une interface plus haut-niveau :

- `otawa::xengine::Xdd` : classe enveloppe autour d'un nœud XDD, permettant de manipuler les opérations de manière opaque sans interagir directement avec les nœuds ou les gestionnaires.
- `otawa::xengine::StandardXddManager` : gestionnaire standard qui combine un `NodeManager` et un `OpsManager`.
- `otawa::xengine::XddMatrix` : permet de manipuler des matrices basées sur des XDD (utile pour des calculs de relations).

Exemple d'utilisation

```cpp
// Exemple d'utilisation de Xdd
#include <otawa/xdd/XddManager.h>
using namespace otawa::xengine;

StandardXddManager man;

auto x = Xdd(man.nMan(), 3);  // cree un XDD pour la variable 3
auto y = Xdd(man.nMan(), 1);  // cree un XDD pour la variable 1
auto z = x + y;               // effectue une operation d'union (ou
    autre)
```

3.0.0.1 Remarque 1 :

Le code pourrait être simplifié. Le fait de devoir passer explicitement un `OpsManager` est lourd.

3.0.0.2 Remarque 2 :

Il n'y a pas de moyen simple de créer un XDD à partir d'un événement sans redescendre à la couche `Node`, ce qui complique l'usage direct.

3.0.0.3 Remarque 3 :

Les feuilles du XDD ont un index de variable égal à `-1` (converti en entier non signé), ce qui peut prêter à confusion. Pourquoi ne pas avoir simplement utilisé 0, déjà réservé pour les feuilles dans la fonction `index()` ?

4 Codage et gestion des événements

Les XDD utilisent des entiers naturels pour représenter les événements. Ainsi, il est nécessaire de maintenir une correspondance explicite entre les événements OTAWA (structure logique) et leur code entier (représentation dans le XDD). L'ordre des entiers a une incidence directe sur les performances du XDD car il influence sa structure.

La classe `XddEdgeVarsManager` est responsable de cette correspondance. Elle associe à chaque événement OTAWA un objet `XddEvent` qui contient :

— l'événement OTAWA,
— l'arête du graphe de flot de contrôle (`CFG`) où se produit l'événement,
— le code entier utilisé dans le XDD.

Cette classe est alimentée automatiquement en appelant la fonction `collectEvents()` sur un objet supportant les propriétés OTAWA.

4.0.0.1 Intégration OTAWA :

Tout cela est intégré à la fonctionnalité `XDD_EVENT_FEATURE` via la propriété `XDDVERS_MANAGER`. La collecte est réalisée par le processeur `XddEventsGenerator`.

5 Structure du code d'événement

Un code d'événement est un entier 64 bits découpé en trois parties :

— Bits `[5..0]` : génération,
— Bits `[31..6]` : identifiant d'événement,
— Bits `[63..32]` : identifiant du bloc de base (BB) au niveau de la collection de CFG.

La classe `XddVarHelper` fournit des fonctions utilitaires pour extraire chacune de ces parties.

5.0.0.1 Remarque :

La classe `XddGeneration` peut être utilisée pour gérer les informations de génération indépendamment.

Conclusion

Le système `otawa-xdd` permet une manipulation avancée et symbolique des événements de l'analyse statique OTAWA, à l'aide de structures efficaces comme les XDD. Néanmoins, la complexité de son code et de sa gestion des événements impose une phase de compréhension approfondie, notamment des relations entre événements, identifiants et représentation dans le graphe.