7.2 Termination Criteria for Graph Rewrite Systems

Prof. Dr. Uwe Aßmann
Softwaretechnologie
Technische Universität Dresden
Version 10-0.1, 20.11.10

1) EARS
2) AGRS
3) SGRS
4) XGRS
Problems with GRS

- With graph rewriting, there are some problems:
  - termination: when does a GRS terminate for a start graph?
    - Idea: identify a termination graph
  - non-convergence: when does a GRS deliver a deterministic solution (unique normal form)?
    - Idea: unique normal forms by rule stratification
Additive Termination

- Identify a termination (sub-)graph
- Conditions in the additive case:
  - nodes of termination (sub-)graph are not added (remain unchanged)
  - its edges are only added
- If the termination graph is complete, the system terminates
- This leads to the definition of **AGRS (edge-Accumulative Graph Rewrite Systems)**.
Example: Subexpressions

"Find all subexpressions which are reachable from a statement"

\[
\text{Exprs}(\text{Stmt}, \text{Expr}) : - \text{Child}(\text{Stmt}, \text{Expr}).
\]

\[
\text{Exprs}(\text{Stmt}, \text{Expr}) : - \text{Child}(\text{Stmt}, \text{Expr2}), \text{Descendant}(\text{Expr2}, \text{Expr}).
\]

\[
\text{Descendant}(\text{Expr1}, \text{Expr2}) : - \text{Child}(\text{Expr1}, \text{Expr2}).
\]

\[
\text{Descendant}(\text{Expr1}, \text{Expr2}) : - \text{Descendant}(\text{Expr1}, \text{Expr3}), \text{Child}(\text{Expr3}, \text{Expr2}).
\]

Features:
- terminating, strong confluent
- convergent (unique normal form)
- recursive

Why do such graph rewrite systems terminate?
Execution of „Reachable Subexpressions“

Diagram:
- Assign
  - Expr
    - Const
      - 1
    - Var
      - X
  - Plus
    - Exprs
      - 1
      - X
Execution of „Reachable Subexpressions“

Diagram:

- **Assign**
- **Expr**
- **Plus**
- **Const**
- **Var**
- **1**
- **X**

Arrows indicate:
- **Child**
- **Exprs**
- **Descendants**
Execution of „Reachable Subexpressions“
Execution of „Reachable Subexpressions“

Diagram showing the execution process with nodes labeled as Assign, Expr, Plus, Const, Var, 1, and X.
EARS are Simple AGRS

- A subclass of edge-accumulative graph rewrite systems are EARS (Edge addition rewrite systems).
  - They can be used for the construction of graphs
  - For model and program analysis

- **terminating**: noetherian on finite lattice of subgraphs

- **strongly confluent**: direct derivations can always be interchanged.

- **congruent**: unique normal form
Data-flow Analysis with EARS

- Every distributive data flow problem on finite-height powerset lattices can be represented by an EARS
  - defined/used-data-flow analysis
  - partial redundancies
  - local analysis and preprocessing:
- EARS work for other problems which can be expressed with DATALOG-queries
  - equivalence classes on objects
  - alias analysis
  - program flow analysis
7.2.1 Additive GRS (AGRS)
"Allocate a register object for every subexpression of a statement which has a result and link the expression to the statement"

if Exprs(Stmt, Expr), HasResult(Expr)
then
  ObjectExprs(Stmt, Expr),
  RegisterObject := new Register;
  UsedReg(Expr, RegisterObject)
;
Features: terminating
Assign

Expr

Plus

Const

Var

1

X
Edge-accumulative Rules and AGRS

- A GRS is called *edge-accumulative* (an AGRS) if
  - all rules are edge-accumulative and
  - no rule adds nodes to the termination-subgraph nodes of another rule.

- Edge-accumulative rules are defined on label sets of nodes and edges in rules

- Criterion statically decidable
The Termination Subgraph of the Examples

- Collection of subexpressions:
  \[ T = (\{\text{Stmt,Expr}\}, \{\text{Exprs, Descendant}\}) \]

- Allocation of register objects:
  \[ T = (\{\text{Proc,Expr}\}, \{\text{ObjectExprs}\}) \]
Subtractive Termination (Subtractive GRS)

- Conditions in the subtractive case:
  - the nodes of the termination subgraph are not added (remain unchanged)
  - its edges are only deleted
- If the termination subgraph is empty, the system terminates
- Results in:
  - edge-subtractive GRS (ESGRS)
  - subtractive GRS (SGRS)
- AGRS, SGRS make up XGRS (eXhaustive Graph Rewrite Systems)
Constant Folding as Subtractive GRS

```
Const + Const
  Const
  1
  2
```

```
Const
  3
```
Peephole Optimization as Subtractive XGRS

Diagram:

- **Plus**
  - **Var** → **X**
  - **Const** → **1**
  - *next*

- **Incr**
  - **X**

- **IncrIncr**
  - **Var** → **X**
The Nature of XGRS

- All redex parts in the termination-subgraph of the host graph are reduced step by step.
- The termination-subgraph is either completed or consumed.
  - Edge-accumulative systems may create new redex parts in the termination-subgraph, but there will be at most as many of them as the number of edges in the termination-subgraph.
  - Subtractive systems do not create sub-redexes in the termination-subgraph but destroy them.
- XGRS can only be used to specify algorithms which perform a finite number of actions depending on the size of the host graph.