Canonical Form for Intermediate Representation

Marcel Broeken
mhbroeke@cs.uu.nl

November 16, 2001

Contents

- Introduction
- Canonical trees
- Basic blocks
- Traces
- Summary

TREE language

- ca-non-i-cal: reduced to the simplest or clearest schema possible
- TREE language matches machine capabilities
  - But not always

Mismatches

- CJUMP
  - Can jump to two labels
- ESEQ
  - Order becomes an issue
- CALL
  - Order becomes an issue
  - Problems when writing to RV-register

Rewriting

- Tree without mismatches
- Three steps:
  - Canonical trees
  - Basic blocks
  - Traces

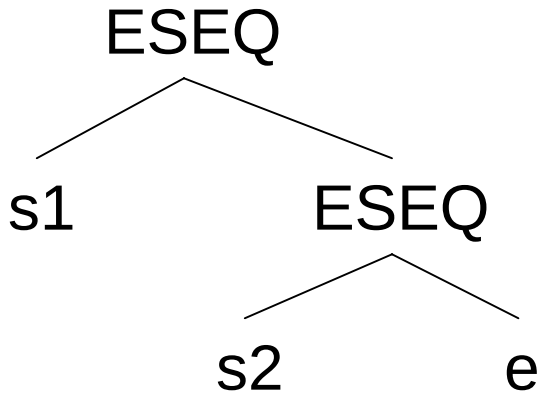
Canonical trees

- Definition:
  - No SEQ or ESEQ
  - Parent of each CALL is:
    - EXP(...)
    - MOVE(TEMP(t), ...)
- Three steps:
  - Eliminate ESEQ
  - Move CALL to top level
  - Eliminate SEQ

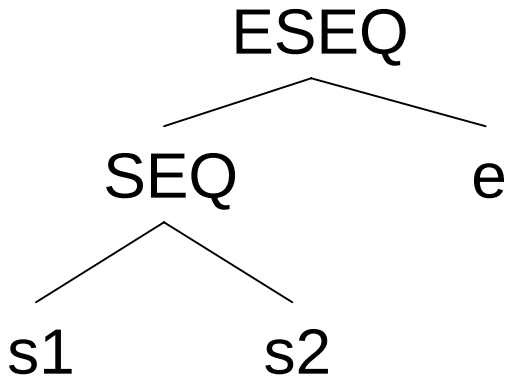
Eliminate ESEQ

- Move ESEQ to top level
  - ESEQ can become SEQ
- Identities on trees (next slide)

Identity 1

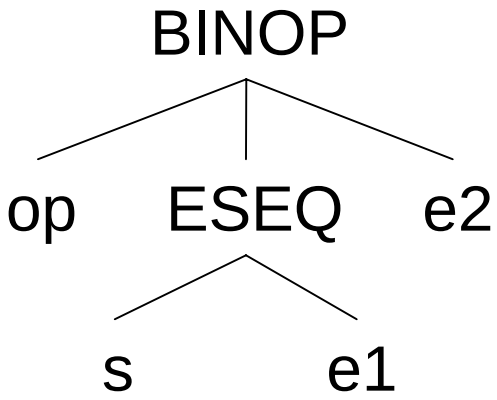


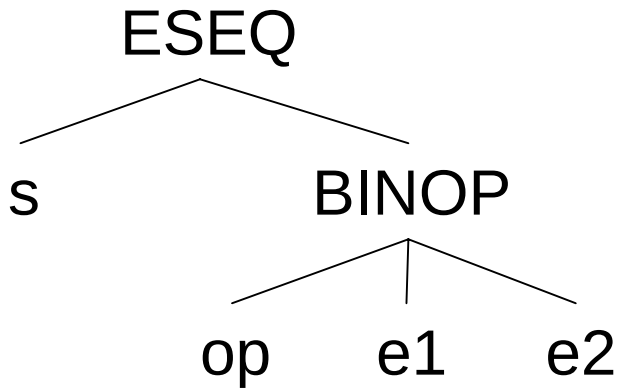
ESEQ(s1, ESEQ(s2, e))



ESEQ(SEQ(s1, s2), e)

Identity 2

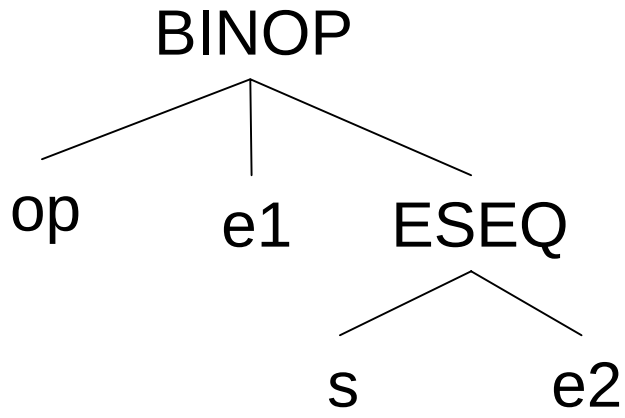


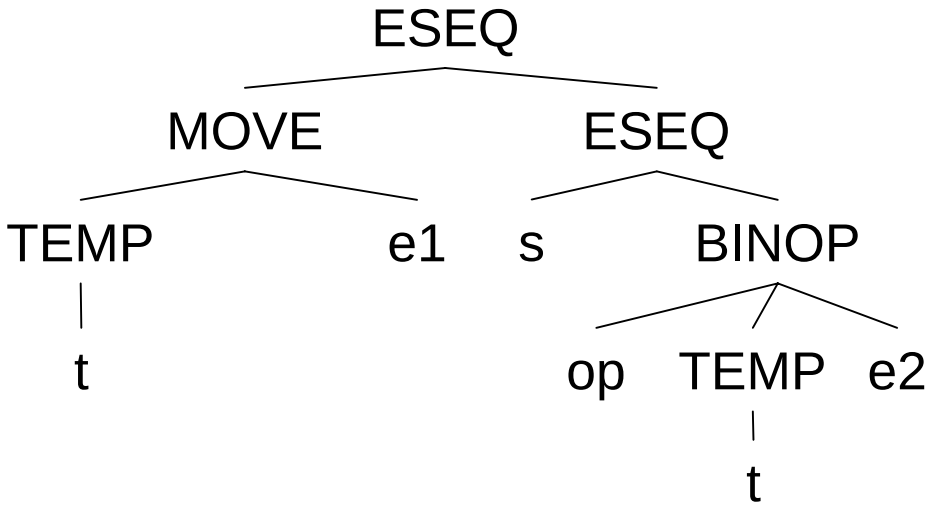


BINOP(op, ESEQ(s, e1), e2)
MEM(ESEQ(s,e1))
JUMP(ESEQ(s, e1))
CJUMP(op, ESEQ(s, e1), e2, l1, l2)

ESEQ(s, BINOP(op, e1, e2))
ESEQ(s, MEM(e1)
SEQ(s, JUMP(e1))
SEQ(s, CJUMP(op, e1, e2, l1, l2))

Identity 3





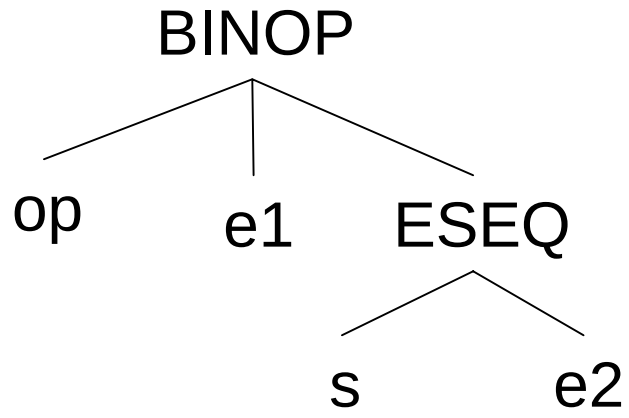
BINOP(op, e1, ESEQ(s, e2))

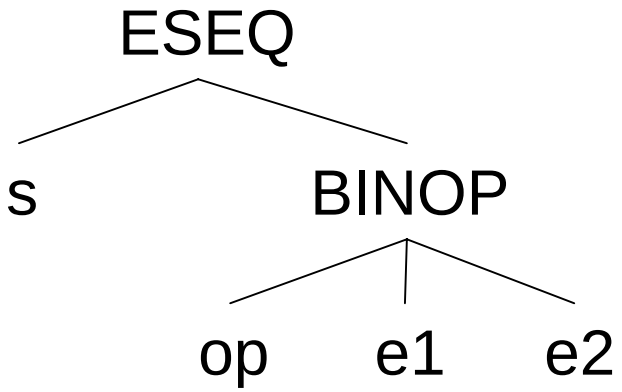
ESEQ(MOVE(TEMP t, e1),
ESEQ(s, BINOP(op, TEMP t, e2)))

CJUMP(op, ESEQ(s, e1), e2, l1, l2)

SEQ(MOVE(TEMP t, e1),
SEQ(s, CJUMP(op, TEMP t, e2,l1,l2)))

Identity 4





BINOP(op, e1, ESEQ(s, e2))
CJUMP(op, e1, ESEQ(s, e2), l1, l2)

ESEQ(s, BINOP(op, e1, e2))
SEQ(s, CJUMP(op, e1, e2, l1, l2))

Commute

- Constants commute with any statement
- Empty statement commutes with any expression
- Anything else doesn't commute

General rewriting rules

- Input: expression-list
- Output: pair (statement, expression-list)
  - Statement of the ESEQ
  - Expressions left of ESEQ that don't commute

Moving CALLs to top level

- TREE allows CALLs as sub expressions
- Implementation CALL:
  - Result in TEMP(RV)
- Temporary results can be overwritten
- Solution: use fresh temporary registers

Rewrite rule

- CALL(fun, args) $\rightarrow$ ESEQ(MOVE(TEMP(t), CALL(fun, args)), TEMP(t))
- Infinite loops are avoided
- MOVE(TEMP(t), CALL(fun, args)) is recognized

Eliminate SEQ

- After rewrite rules:
  - All SEQ-nodes near the root
- SEQ(SEQ(a, b), c) $\rightarrow$ SEQ(a, SEQ(b, c))
  - SEQ(s_1, SEQ(s_2, ..., SEQ(s_{n-1}, s_n), ...))
  - s1, s2, ..., s_{n-1}, s_n

Taming conditional branches

- No equivalent of CJUMP in instruction set
- Use rewrite rules:
  - CJUMP(con, l_t, l_f) followed by LABEL(l_f)
- Use basic blocks and traces

Basic blocks

- Analyse control flow
- Control flow:
  - Sequencing of instructions in a program, ignoring data values
- Basic block:
  - First statement is a LABEL
  - Last statement is a (C)JUMP
  - There are no other LABELs or (C)JUMPs

Making basic blocks

- Scan sequence of statements
- LABEL $\rightarrow$ begin basic block
- (C)JUMP $\rightarrow$ end basic block

Traces

- Order of basic blocks is free
  - CJUMP(con, l_t, l_f) followed by LABEL(l_f)
- Use traces
- Trace:
  - Sequence of statements that could be consecutively executed during the execution of the program

Generation of traces

Put all the blocks of the program into a list Q.

while Q is not empty

    Start a new trace, call it T

    Remove the head element b from Q.

    while b is not marked

        Mark b; append b to the end of T.

        Examine the successors of b

        if there is any unmarked successor c

        b $\leftarrow$ c

    End T

Finishing up

- CJUMP(con, l_t, l_f) followed by its true label
  - Negate con and switch l_t and l_f
- CJUMP(con, l_t, l_f) followed by no label
  - Rewrite in three statements:
    - CJUMP(con, l_t, l'_f)
    - LABEL l'_f
    - JUMP(NAME l_f)

Optimal traces

- Frequently executed code in one trace
  - Body of a loop
- Minimizes number of jumps
- Simplifies optimisation
  - Register allocation
  - Instruction scheduling

Summary

- Remove mismatches with rewrite rules
- Three steps:
  - Canonical tree
    - Eliminate ESEQ
    - Move CALLs to top level
    - Eliminate SEQ
  - Basic blocks
  - Traces

Questions