SMT Final Project

Program Verification by Underapproximation of Weakest Precondition and Strongest Postcondition

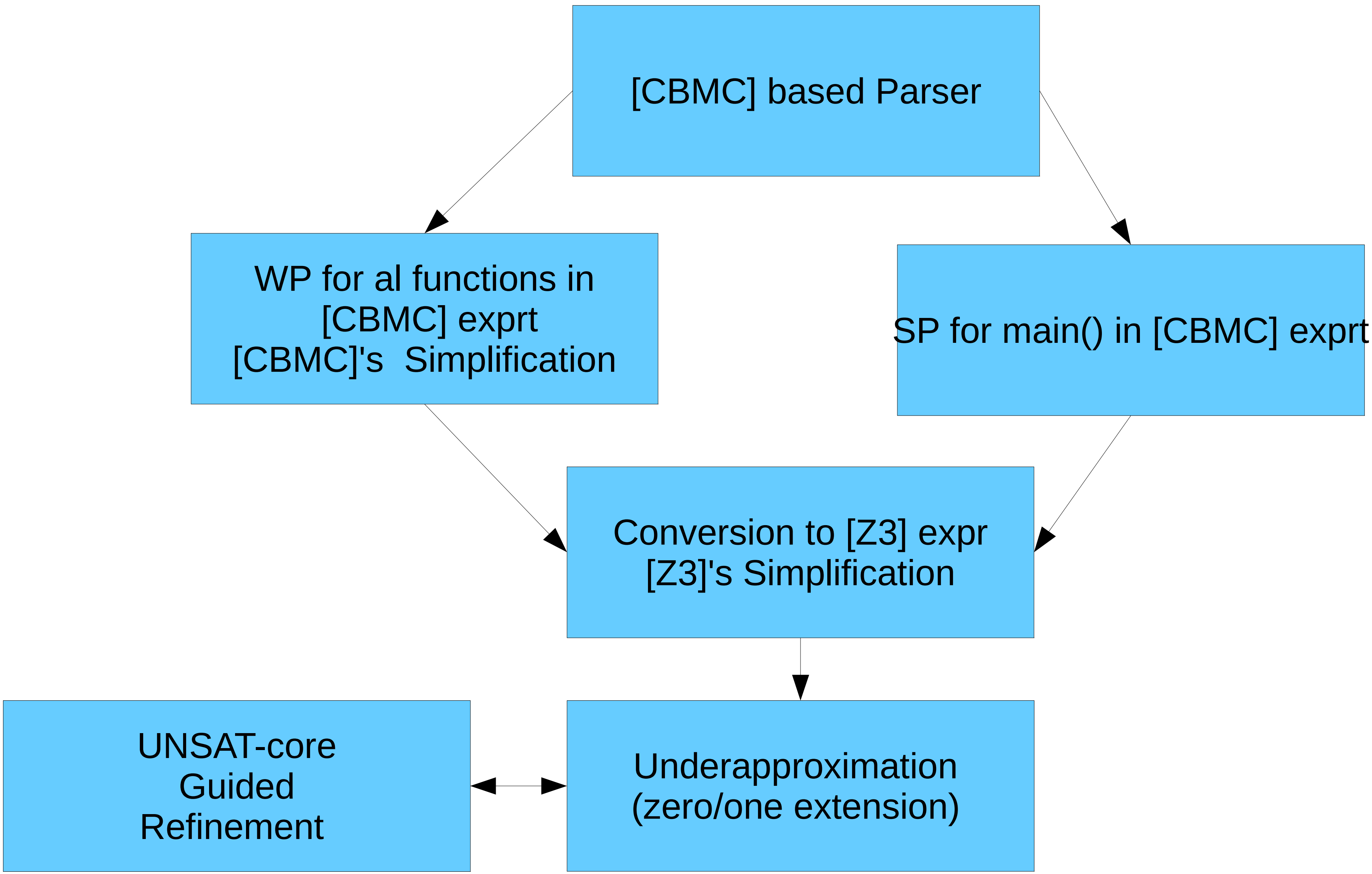
Extension of WP to Function Summaries

Work done as part of *Software Verification Using SMT Solvers* course under Mandayam Srivas by Sumanth Prabhu S and Mohammad Afzal.

Overview

- Computation of WP and SP for straight line C programs with integer type.
- Verification by underapproximation of size of program variables.
- Refinement of underapproximation using unsat core.
- Extension to functions with no parameter or return value by on-the-fly function summary computation.

Design


[CBMC] based Parser
WP for al functions in [CBMC] exprt [CBMC]'s Simplification
SP for main() in [CBMC] exprt
Conversion to [Z3] expr [Z3]'s Simplification
UNSAT-core Guided Refinement
Underapproximation (zero/one extension)

Underapproximation

- Additional constraints are added to restrict the size of variables.
- Zero Extension
  - $P' = P \wedge (e_x^i \rightarrow (x[n..w] = 0)$

    $\wedge (e_y^i \rightarrow (y[n..w] = 0) \ldots$
- One Extension
  - $P' = P \wedge (e_x^i \rightarrow (x[n..w] = \text{repeat } (n\text{-}w)\ 1\text{'s})$

    $\wedge (e_y^i \rightarrow (y[n..w] = \text{repeat } (n\text{-}w)\ 1\text{'s}) \ldots$
- We have modified the constraint formula from [Brummayer et al.]

Underapproximation

- No flattening.
  - x[n..w] = logical_right_shift(x , w).
  - (n-w) 1's = logical_right_shift(~0, w) where ~0 is bitvector complementation.
- If any of e_x^i are found in unsat core:
  - e_x^i dropped as assumption for next iteration.
  - Width of corresponding variable is incremented and e_x^{i+1} is added.
- We are incrementing width exponentially with initial width 4 (i.e. 4,8,16 and 32).

Function Summary

Example 1

foo:
x => x + 1
wp' = true

bar:
x => x
wp' = (x > 10)

Computed wp

```
int x, y, z;

void foo()
{
  x = x + 1;
}

void bar()
{
  assert(x > 10);
}
```

Example from [Madhukar et al.]

```
int main()
{
  if (x < 0) {
    foo();
    foo();
    bar();
  }
}
```

(x < 0)
&& ((x+1) + 1 > 10)
&& true
|| !(x<0)

(x < 0)
&& ((x+1) > 10)
&& true
|| !(x<0)

(x<0)
&& (x > 10)
&& wp(foo(), true)
|| !(x<0)

(x<0)
&& wp(bar(), true)
|| !(x<0)

wp(main(), true)

Function Summary

Example 2

```
foo:
wp' =
( y < 0 && 1 + 3*y > 0
  || !(y < 0) && 1 + 3*x > 0)

X => (y < 0) ? (2*y + 1)
             :  1 + x

Y => !(y<0) ?  (2*x + 1)
             : y
```

```
int x, y, z;

void foo()
{
  if (y < 0) {
    x = y + y;
  } else {
    y = x + x;
  }
  x = x + 1;
  assert((x + y) > 0);
}
```

```
int main()
{
  __CPROVER_assume((x == 64)
                   && (y == 32));

  foo();

  z = x + y;

  assert(z == 193);
}
```

```
(y < 0 ? 2*y + 1 : 1 + x) + (!(y < 0) ? 2*x : y) == 193
&& (y < 0 && 3*y + 1 > 0 || !(y < 0) && 1 + 3*x > 0)
 || !(x == 64) || !(y == 32))
```

Function Summary

Algorithm

```
wp (P, phi)
  for each stmt in P do
    if stmt = (assert psi)
      phi := phi && psi

    else if stmt = (assume psi)
      phi := psi -> phi

    else if stmt = (x := psi)
      phi := phi [x := psi]

    else if stmt = (if guard then stmts1 else stmt2)
      phi := (guard && wp(stmts1, phi)) || (!guard && wp(stmt2, phi))

    else if stmt = f()
      if (!wp' of f available)
        wp' := wp(f, true)
      phi := phi[x := x', y := y'...] && wp'
      //here x', y', etc. are substitution values
```

Reference for wp without funtion call: [Lieno]

Strongest Postcondition

Algorithm

```
sp (P, phi)
  for each stmt in P do
    if stmt = (assert psi)
      phi := phi → psi

    else if stmt = (assume psi)
      phi := psi && phi

    else if stmt = (x := psi)
      phi := exists t (x==psi[x:=t] && phi[x:=t])

    else if stmt = (if guard then stmts1 else stmt2)
      phi := sp(stmts1, phi && guard) || sp(stmt2, phi && !guard)
```

Reference: [Gordon et al.] and ProgramVerifFloydHoardSlides.pdf

References

- [Brummayer et al.] Effective Bit-Width and Under-Approximation - Robert Brummayer and Armin Biere.
- [Madhukar et al.] Compositional Safety Refutation Techniques - Kumar Madhukar, Peter Schrammel, and Mandayam Srivas.
- [Lieno] Efficient weakest preconditions - K. Rustan M. Leino, weakest conservative precondition.
- [Gordon et al.] Forward with Hoare – Mike Gordon and Helene Collavizza.
- [CBMC] http://www.cprover.org/svn/cbmc/releases/cbmc-4.5/
- [Z3] https://github.com/Z3Prover/z3