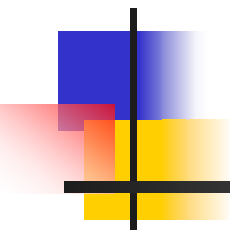


An Enhanced Assertion Checking Environment for C Programs



Navneet Kataria
02305406

Under the guidance of
Prof. S. Ramesh



Outline

- Introduction.
- Motivation.
- ACE, an assertion checking environment.
- Introduction to PVS.
- Future work.



Introduction

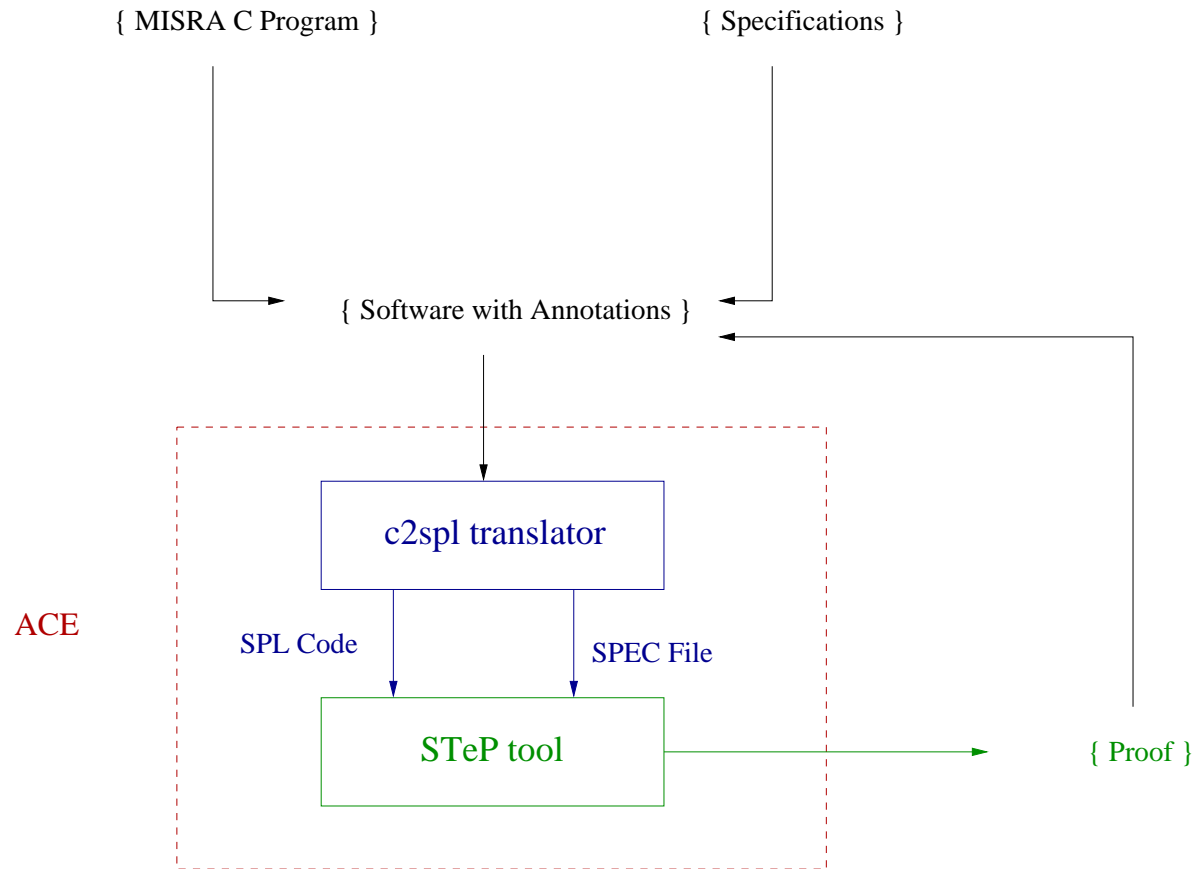
- Importance of formal verification of systems.
- Approaches for formal verification.
 - Model-checking.
 - Theorem proving.



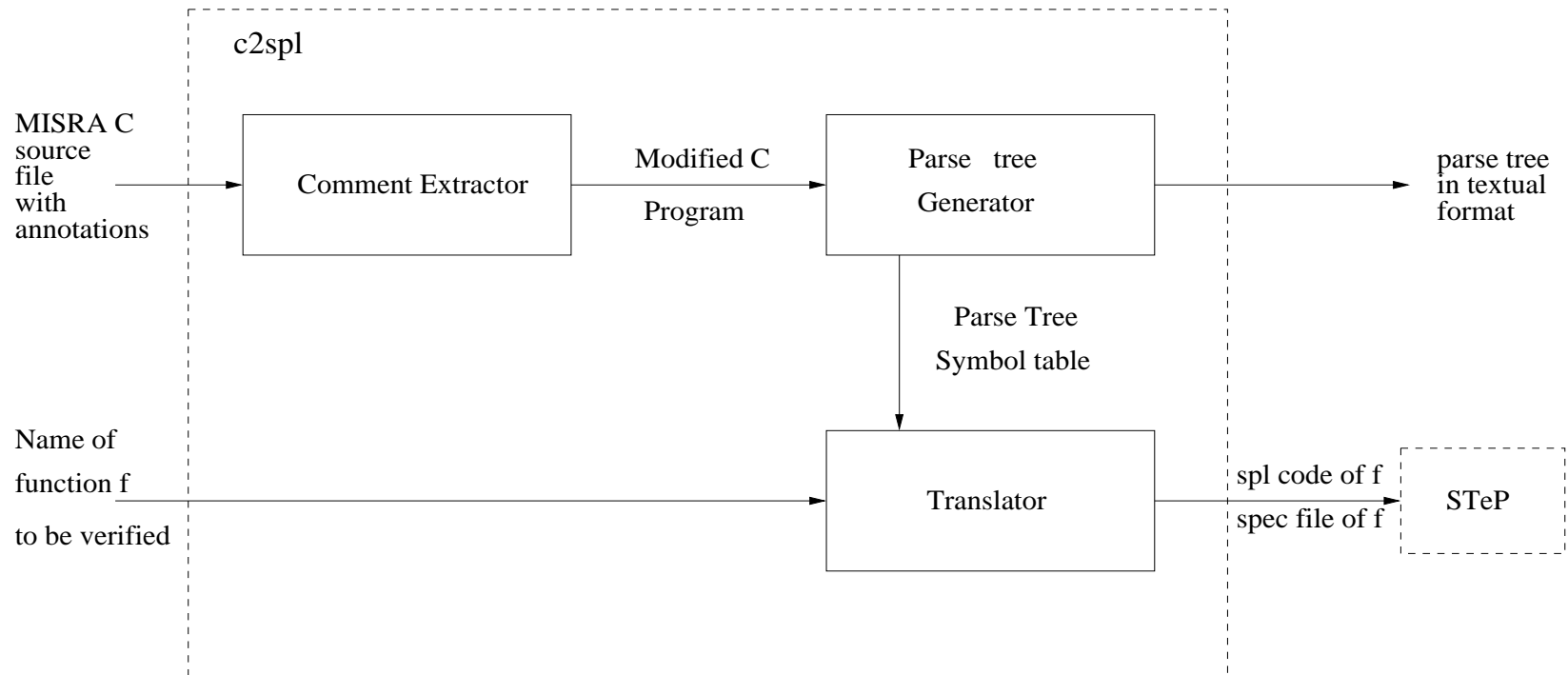
Motivation

- Need of an assertion checking environment.
 - Non-availability of C language program verifier.
- ACE – first C language program verifier.

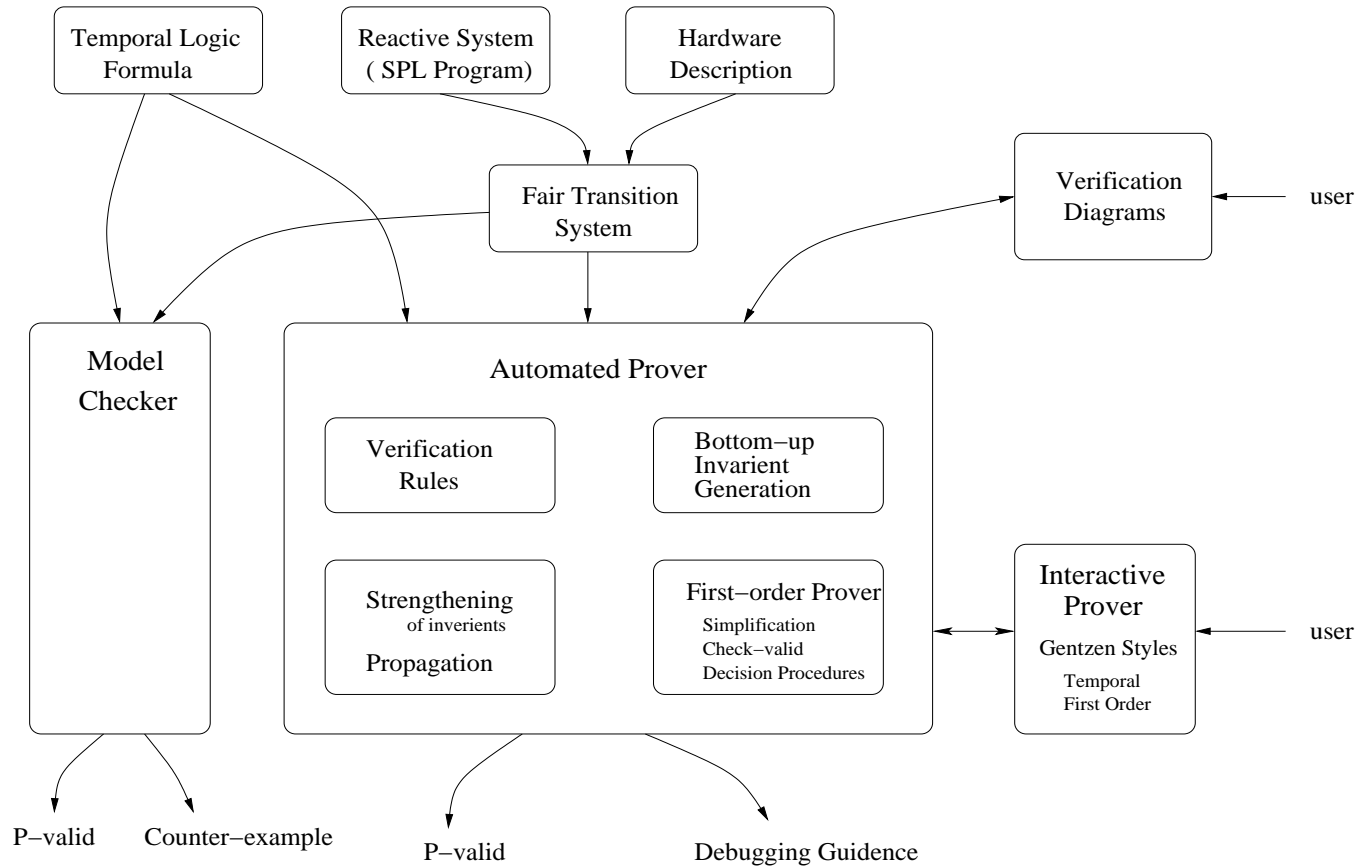
ACE : an Assertion Checking Environment



The C2spl Translator



The STeP





Example

```
int main()
{
int n=5;
int result=0;
int i=1;
int temp=0;
int temp1=0;

/*pre (n>=1) end*/
```

```
while(i<=n)
{
temp1=i;
temp=result;
result=temp+i;
i=temp1+1;
}
/*post result=(n * (n+1) /
2) end*/
}
```



Example (Cont..)

- Generated SPL code

% Declarations

%-----

main :: [
 out RET_main : int

 local n : int where n = 5
 local result : int where result = 0
 local i : int where i = 1
 local temp : int where temp = 0
 local temp1 : int
 where temp1 = 0

pre1 : skip;

while ((i <= n)) do
 [
 temp1 := i;
 temp := result;
 result := (temp + i);
 i := (temp1 + 1)
];

post2 : skip;

RET_main := result
]



Example (Cont..)

- Generated Specification File

SPEC

AXIOM pre1 : pre1 ==> (n >= 1)

PROPERTY post2 :

post2 ==> result = (n * (n + 1) / 2)



Comments on ACE

- Good Things.
 - Already used in industrial projects.
 - First successful effort.
- Restrictions.
 - No support for automated checks.
 - Not powerful.
 - STeP.
 - Lack of support for concurrent program verification.



Prototype Verification System

- Theorem prover.
- Sophisticated type system.
- Expressive specification language.
 - Theories.
 - Functions.
 - Theorem and axioms.
 - Types.



Example

Stacks [t:TYPE +] :THEORY

BEGIN

stack : TYPE+

s : VAR stack

empty : stack

nonemptystack?(s) : bool = s /= empty

push : [t, stack → (nonemptystack?)]

pop : [(nonemptystack?) → stack]

top : [(nonemptystack?) → t]



Example (Cont..)

$x, y : \text{VAR } t$

push_top_pop : AXIOM

$\text{nonemptystack?}(s) \text{ IMPLIES } \text{push}(\text{top}(s), \text{pop}(s)) = s$

pop_push : AXIOM $\text{pop}(\text{push}(x,s)) = s$

pop2push2 : THEOREM

$\text{pop}(\text{pop}(\text{push}(x, \text{push}(y, s)))) = s$

END Stacks



PVS Theorem-prover

- Goal reduction approach.
- Powerful atomic steps.
- Type correctness conditions (TCCs).
- Prelude library.
- Strategies or tactics.



Conclusion and Future Work

- ACE is not powerful.
- New assertion checking environment.
- PVS as a back-end verifier.
- Concurrent program verification.



Thank You
