

Certificate Translation

***Gilles Barthe, Cesar Kunz, Benjamin Gregoire, Tamara Rezk
INRIA Sophia Antipolis***

***Workshop on PCC 2006
August 11, Seattle***

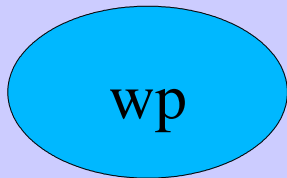
Motivation

- Certifying compilation aims to support automatic verification of safety properties
- Certificates for complex properties often require the use of interactive code verification
- The objective of this work is to bring the benefits of source code verification to code consumers

Setting for certificate translation

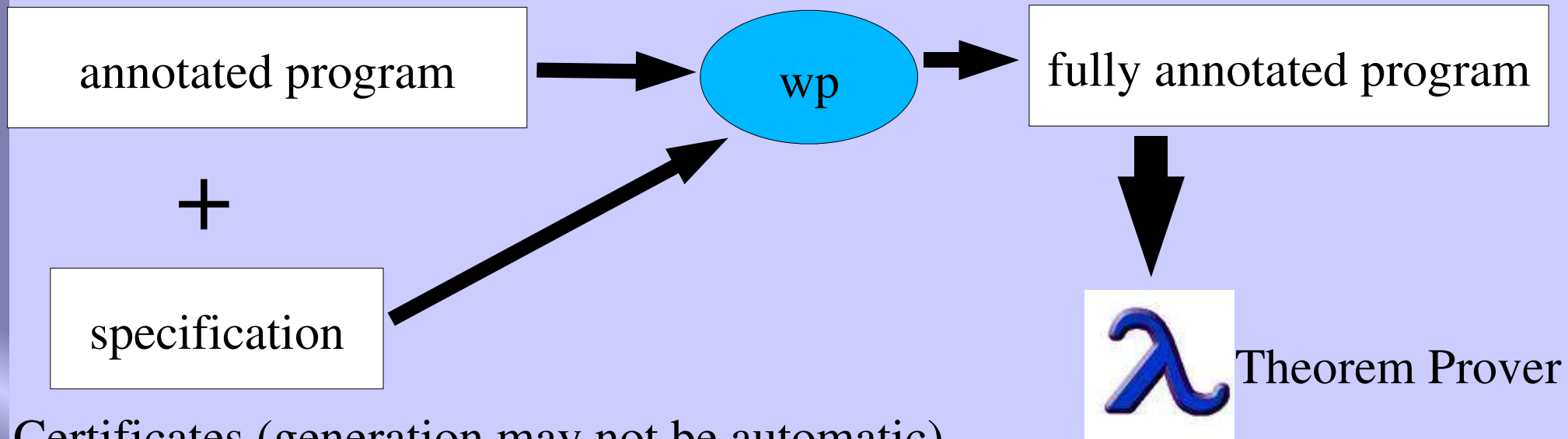
Programs are equipped with certificates

- assumption: programs are annotated with loop invariants + precondition and postcondition
- a weakest precondition for instructions (compute preconditions of all instructions)



- Dijkstra-like weakest precondition
- but it does not apply to loops (programs are annotated)

Code Verification

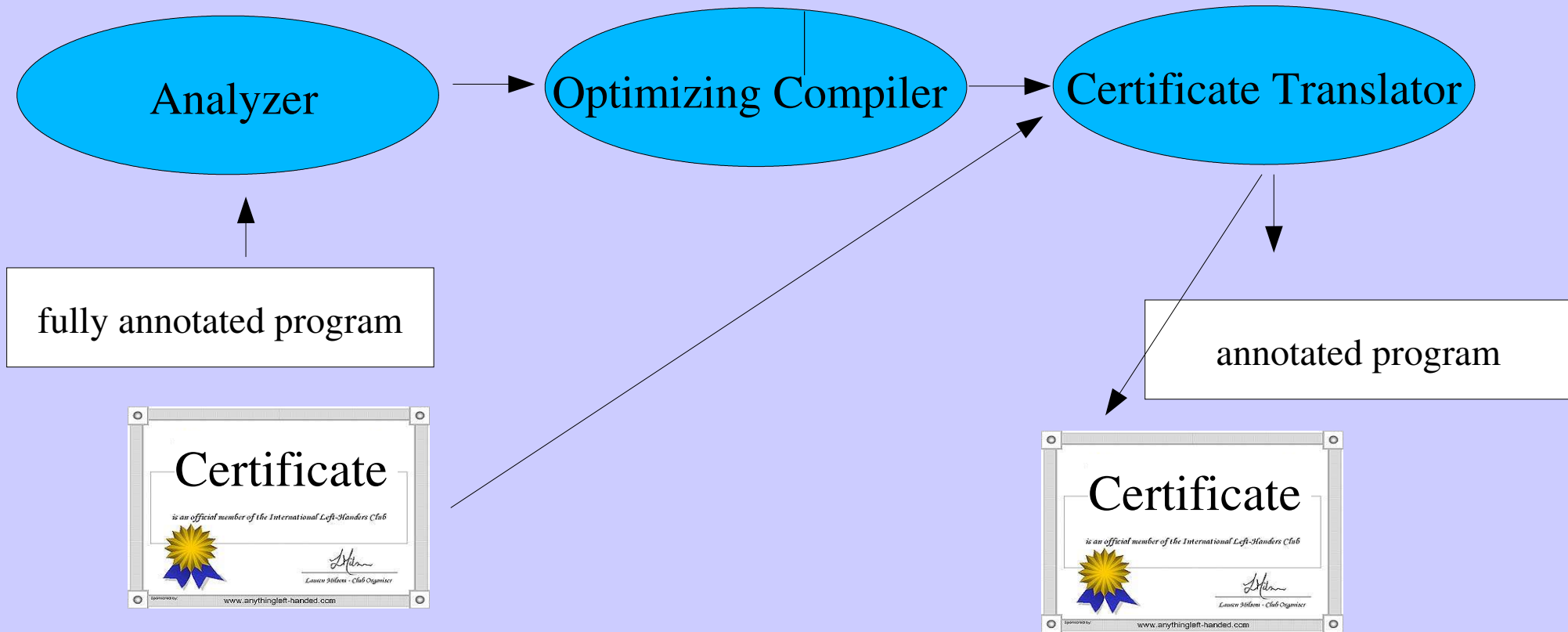


Certificates (generation may not be automatic)

- for every invariant, there is formal proof that the invariant implies the weakest precondition
- there is a formal proof that the given precondition implies the weakest precondition of the first instruction



Certificate Translator

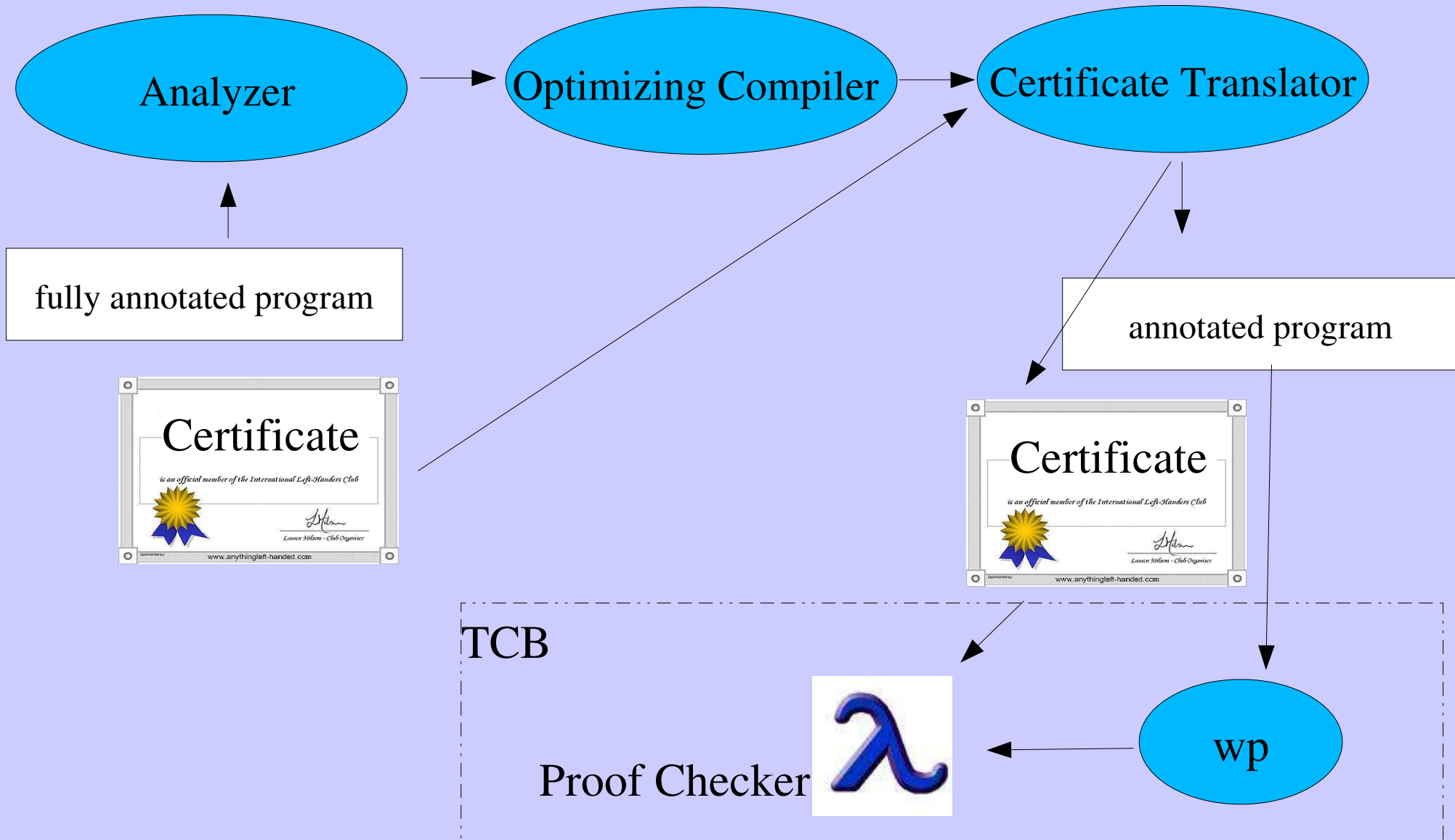


At source level, generation may not be automatic

Certificates at target level, are automatically generated

logic and certificate format may change

Certificate Translator



Certificate translation is not ...

- ◆ certified compilation (encapsulation of compiler definition in the proof and limited to input/output properties)
- ◆ certifying compilation (limited to safety properties)

Certificate translation for non-optimizing compilers

Theorem (Preservation of Proof Obligations)
Source and compiled programs have syntactically equal proof obligations

(Almost PPO for Java)

Certificate translation for optimizing compilers

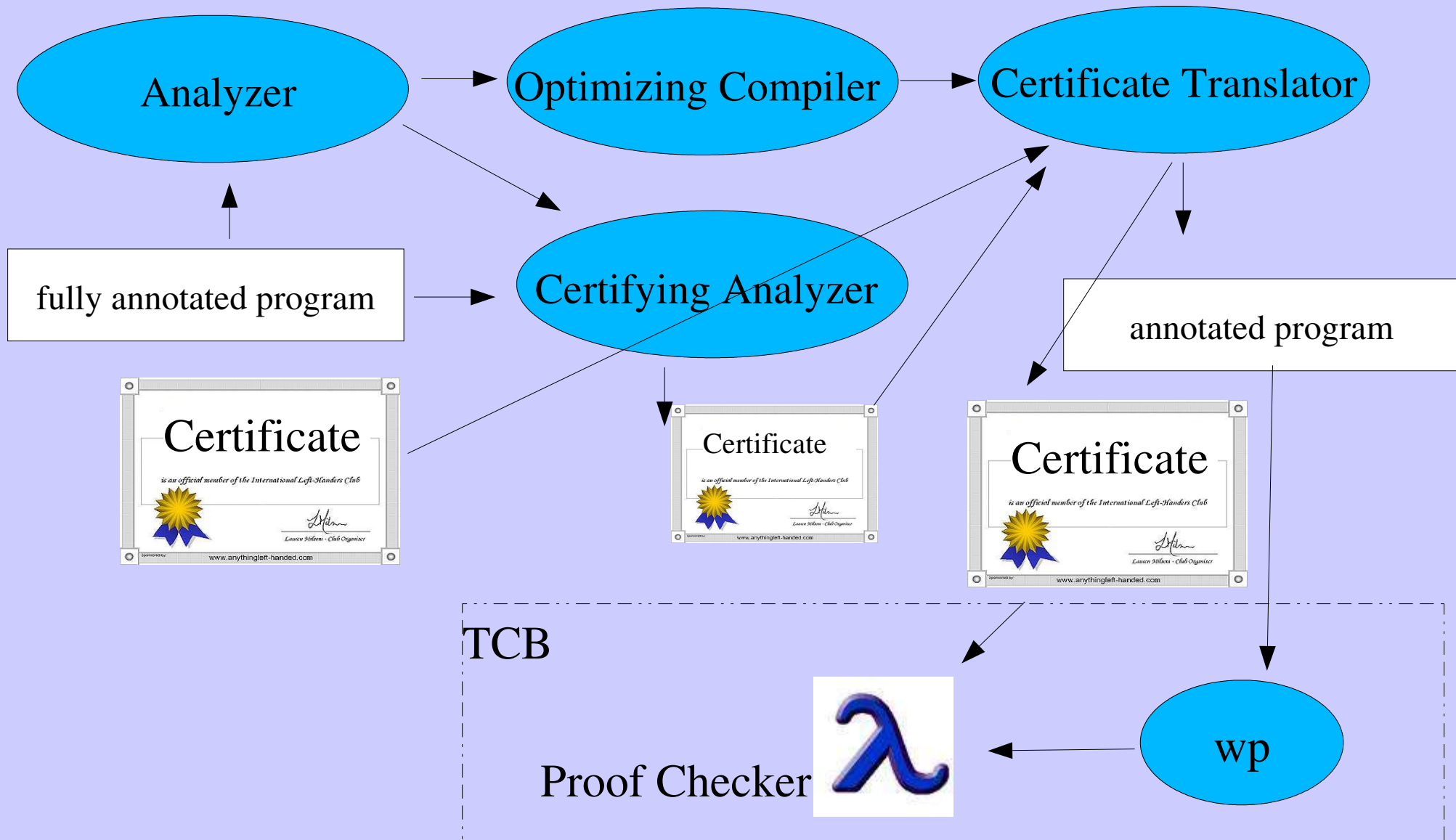
- proofs obligations might not be syntactically preserved
- annotations might need to be modified (e.g. constant propagation)
- certificates for analyzers might be needed (certifying analyzer)
- analyses might need to be modified (e.g. dead variable elimination)

Certificate translation for optimizing compilers

We consider common optimizations at the level of an intermediate RTL language:

- Constant propagation
- Loop induction register strength reduction
- Common subexpression elimination
- Dead register elimination
- Register allocation
- Inlining
- Dead code elimination

Certificate Translator with certifying analyzer



Example

$\{j=0\}$

$i := 0;$

$x := b + i;$

while ($i \neq n$)

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

$i := c + i;$

$j := x * i;$

endwhile

$\{n * b \leq j\}$

annotated program

+

specification

Example

$\{j=0\}$

$\{j = x * 0 \wedge b \leq b \wedge 0 \leq 0\}$

$i := 0;$

$\{j = b+i * i \wedge b \leq b+i \wedge 0 \leq i\}$

$x := b + i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

while ($i \neq n$)

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

$\{x*(c+i) = x * (c+i) \wedge b \leq x \wedge 0 \leq c+i\}$

$i := c + i;$

$\{x*i = x * i \wedge b \leq x \wedge 0 \leq i\}$

$j := x * i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

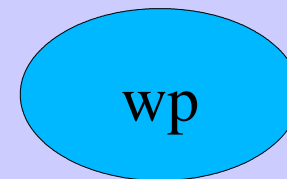
endwhile

$\{n * b \leq j\}$

annotated program

+

specification



No fix point
to calculate



fully annotated program

Example

$\{j=0\}$

$\{j = x * 0 \wedge b \leq b \wedge 0 \leq 0\}$

$i := 0;$

$\{j = b+i * i \wedge b \leq b+i \wedge 0 \leq i\}$

$x := b + i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

while ($i \neq n$)

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

$\{x*(c+i) = x * (c+i) \wedge$

$b \leq x \wedge 0 \leq c+i\}$

$i := c + i;$

$\{x*i = x * i \wedge b \leq x \wedge 0 \leq i\}$

$j := x * i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

endwhile

$\{n * b \leq j\}$

fully annotated program

Proof Obligations

(1) $j = x * i \wedge b \leq x \wedge 0 \leq i \wedge i \neq n \rightarrow$

$x*(c+i) = x * (c+i) \wedge b \leq x \wedge 0 \leq c+i$

(2) $j = x * i \wedge b \leq x \wedge 0 \leq i \wedge i = n \rightarrow$

$n * b \leq j$

(3) $j=0 \rightarrow j = x * 0 \wedge b \leq b \wedge 0 \leq 0$



Theorem Prover

Example

Analyzer

<code>i := 0;</code>	→	<code>(i,0)</code>
<code>x := b + i;</code>	→	<code>(i,0),(x,b)</code>
<code>while (i != n)</code>		
<code> i := c + i;</code>	→	<code>(x,b)</code>
<code> j := x * i;</code>	→	<code>(x,b)</code>
<code>endwhile</code>		

Example

$\{j=0\}$

$\{j = b * 0 \wedge b \leq b \wedge 0 \leq 0\}$

$i := 0;$

$\{j = b * i \wedge b \leq b+i \wedge 0 \leq i\}$

$x := b ;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

while ($i \neq n$)

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

$\{b*(c+i) = x * (c+i) \wedge$

$b \leq x \wedge 0 \leq c+i\}$

$i := c + i;$

$\{b*i = x * i \wedge b \leq x \wedge 0 \leq i\}$

$j := b * i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

endwhile

$\{n * b \leq j\}$

fully annotated optimized program

(1) $j = x * i \wedge b \leq x \wedge 0 \leq i \wedge i \neq n \rightarrow$

$b*(c+i) = x * (c+i) \wedge b \leq x \wedge 0 \leq c+i$

(2) $j = x * i \wedge b \leq x \wedge 0 \leq i \wedge i = n \rightarrow$

$n * b \leq j$

(3) $j=0 \rightarrow j = x * 0 \wedge b \leq b \wedge 0 \leq 0$

Example

$\{j=0\}$

$\{j = b * 0 \wedge b \leq b \wedge 0 \leq 0\}$

$i := 0;$

$\{j = b * i \wedge b \leq b+i \wedge 0 \leq i\}$

$x := b ;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

while ($i \neq n$)

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

$\{b*(c+i) = x * (c+i) \wedge$

$b \leq x \wedge 0 \leq c+i\}$

$i := c + i;$

$\{b*i = x * i \wedge b \leq x \wedge 0 \leq i\}$

$j := b * i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

endwhile

$\{n * b \leq j\}$

fully annotated optimized program

$(1) j = x * i \wedge b \leq x \wedge 0 \leq i \wedge i \neq n \rightarrow$
 $b*(c+i) = x * (c+i) \wedge b \leq x \wedge 0 \leq c+i$



This predicate is not provable,
there is no hypothesis to
conclude $x=b$

Example

$\{j=0\}$

$\{j = b * 0 \wedge b \leq b \wedge 0 \leq 0 \wedge b=b\}$

$i := 0;$

$\{j = b * i \wedge b \leq b+i \wedge 0 \leq i \wedge b=b\}$

$x := b ;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i \wedge x=b\}$

while ($i \neq n$)

$\{j = x * i \wedge b \leq x \wedge 0 \leq i \wedge x=b\}$

$\{b*(c+i) = x * (c+i) \wedge$

$b \leq x \wedge 0 \leq c+i\}$

$i := c + i;$

$\{b*i = x * i \wedge b \leq x \wedge 0 \leq i\}$

$j := b * i;$

$\{j = x * i \wedge b \leq x \wedge 0 \leq i\}$

endwhile

$\{n * b \leq j\}$

The solution is to strengthen the original annotations (invariants), using the results of the analyzer



(x,b)

Example

```
{true}
{b=b}
i := 0;
{ b=b}
x := b ;
{ x=b}
while (i != n)
{x=b}
{true}
  i := c + i;
{true}
  j := b * i;
{true}
endwhile
{true}
```

The results of the analyzer are certified by the certifying analyzer

Certifying Analyzer

Proof Obligations

$x=b \rightarrow true$

$true \rightarrow b=b$



Conclusion

- Certificate translation is a mechanism that allows transferring correctness proofs from source programs to compiled programs
- Prototype in Ocaml: RTL language with arrays and function calls, interface COQ