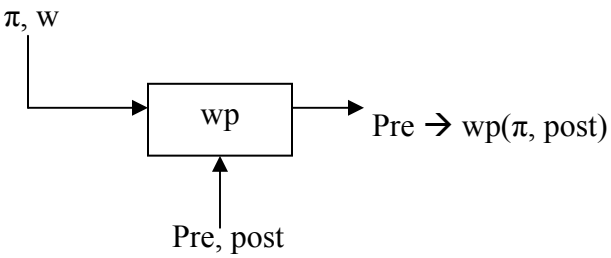
CSE 509 Class note 3/29

Proof carrying code

Pre ➔ wp(π, post)



Pre

S0	Suppose
Assert(P1)	Pre ➔ wp(S0, P1) ∧
S1	P1 → wp(S1, P2) ∧
Assert(P2)	P2 → wp(S2, P3) ∧
S2	P3 → wp(S3, post)
Assert(P3)	----------
S3	Pre ➔ wp(S0;S1;S2;S3, post)
Post	

While(e) {	[P ∧ wp(s,e) → P] ∧
Assert(p);	[¬e → post] ∧
S;	[e → P] ∧
}	[P ∧ wp(s, ¬e) → post]
assert(post)	

L1 :	←———— P → P0
assert(p)	
	←———— P0 …
S1	
	←———— P1 …
S2	
…	…
	←———— P(n-1) …
Sn	
	←———— Pn : x ≠ → post
	∧ x=0 → P
bez x,L1	
→ post	

Wp(bez x,L , post) = $x \neq 0 \rightarrow post \wedge$
$x=0 \rightarrow P \wedge$
$P \rightarrow wp(S1;S2;\ldots;Sn, x \neq 0 \rightarrow post \wedge x=0 \rightarrow P)$

Loop invariants == Programmer work

Tail(l : τ list) Pre = l : τ list
Return l→next; post = ret : τ list

Wp(return l→next , ret : τ list)

L : τ list $\Rightarrow$ l→next : τ list

Assumption
What the assumption implies

m ⊦ l : τ list .
m ⊦ l→next : τ list

➔ If l is a τ list, then l→next is also a τ list

Tail(l : τ list)
0x12345678 = 0; ----------> ①
Return l→next;
Wp(0x12345678=0 ; return l→next , ret : τ list)
= l : τ list ⇒l→next : τ list ----------->②

adding ① does not affect to the wp even though it violates the memory safety.

Wp(r=*p , Q) = ⌈ *p / r ⌉Q
$\wedge$ p : addr
wp(*p = r , Q) = ⌈ r / *p ⌉Q
$\wedge$ p : addr

② becomes
l : τ list ⇒ [l→next : τ list $\wedge$ 0x12345678 : addr]

m ⊦ l : τ list .
m ⊦ l→next : τ list ∧ l : addr ∧ l→data : τ

Theorem prover can prove only safe instructions, so ② will be rejected because T.P doesn't know whether it's safe or not.

	RPC	SFI	PCC
Domain crossing	2	X	X
Argument marshalling	2	1	X
Overhead			Proof overhead
Advantage			Cross domain calls = function calls

Costs of PCC

- programmer effort
- Theorem proving / checking
- Writing interface specs
- Runtime checks
- Exec size ?