



**Whimori**

# 산업제어용 프로그래밍환경과 검증시스템

신승철

한국기술교육대

2009. 7. 9

소프트웨어 무결점 연구센터 워크샵



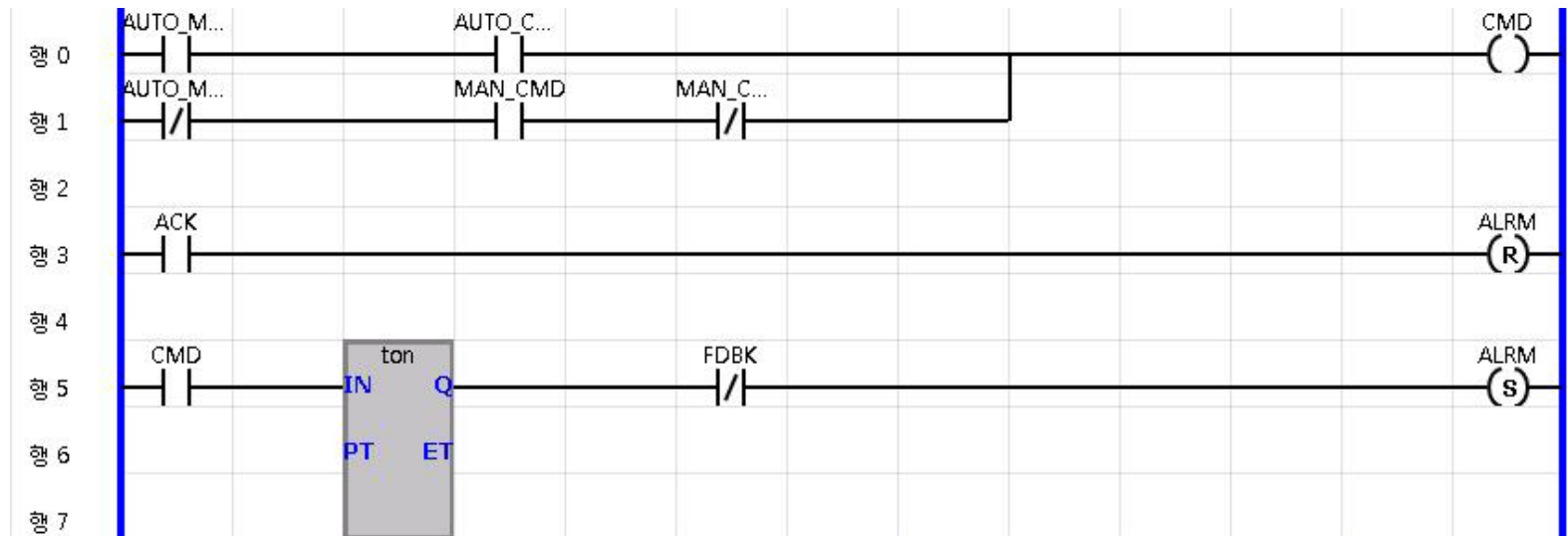
# Background

- PLC(Programmable Logic Controller)
  - a special-purpose microcontroller
  - industrial automation for facilities, equipments, devices
- Development Environments
  - for PLC control programs
  - IsaGraf, Veremiz
  - graphic editors, simulator, target code generators
- IEC61131: PLC standard
  - PLC architecture
  - control programming languages

# PLs for PLC are Street Fighters

- evolved from historical needs
- not driven by modern theory
- no latest abstraction
  - object orientation
  - abstract data types
  - graphical programming environments
- error prone and lower abstract

# IEC61131-3: Ladder Diagram





# IEC61131-3: Structured Text

```
IF R1 THEN
    OFLO := 0; EMPTY := 1; PTR := -1;
    NI := LIMIT (MN:=1, IN:=N, MX:=128); OUT := 0;
ELSIF POP & NOT EMPTY THEN
    OFLO := 0; PTR := PTR-1; EMPTY := PTR < 0;
    IF EMPTY THEN OUT := 0;
    ELSE OUT := STK[PTR];
    END_IF ;
ELSIF PUSH & NOT OFLO THEN
    EMPTY := 0; PTR := PTR+1; OFLO := (PTR = NI);
    IF NOT OFLO THEN OUT := IN ; STK[PTR] := IN;
    ELSE OUT := 0;
    END_IF ;
END_IF ;
```



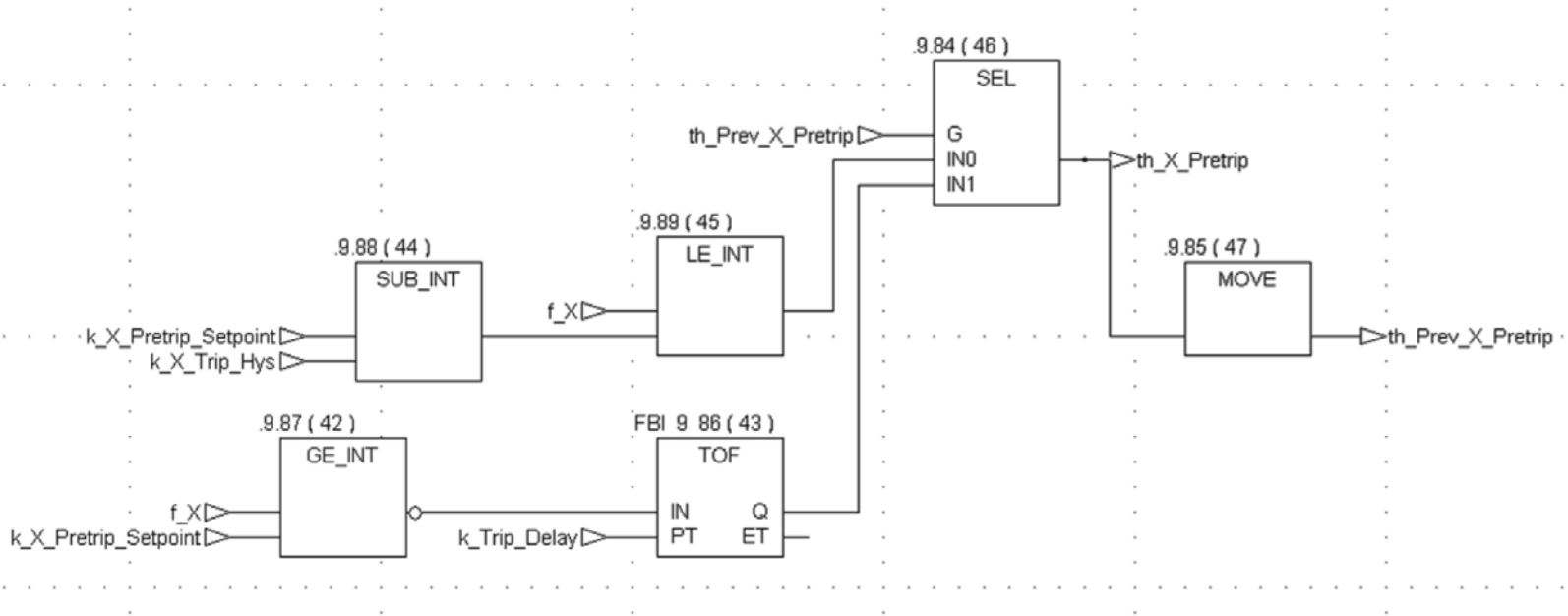
# IEC61131-3: Instruction List

```
POP_STK:    LD    0
            ST    OFLO    (* Popped stack is not overflowing *)
            LD    PTR
            SUB   1
            ST    PTR
            LT    0        (* Empty when PTR < 0 *)
            ST    EMPTY
            JMPC  ZRO_OUT
            LD    STK[PTR]
            JMP   SET_OUT

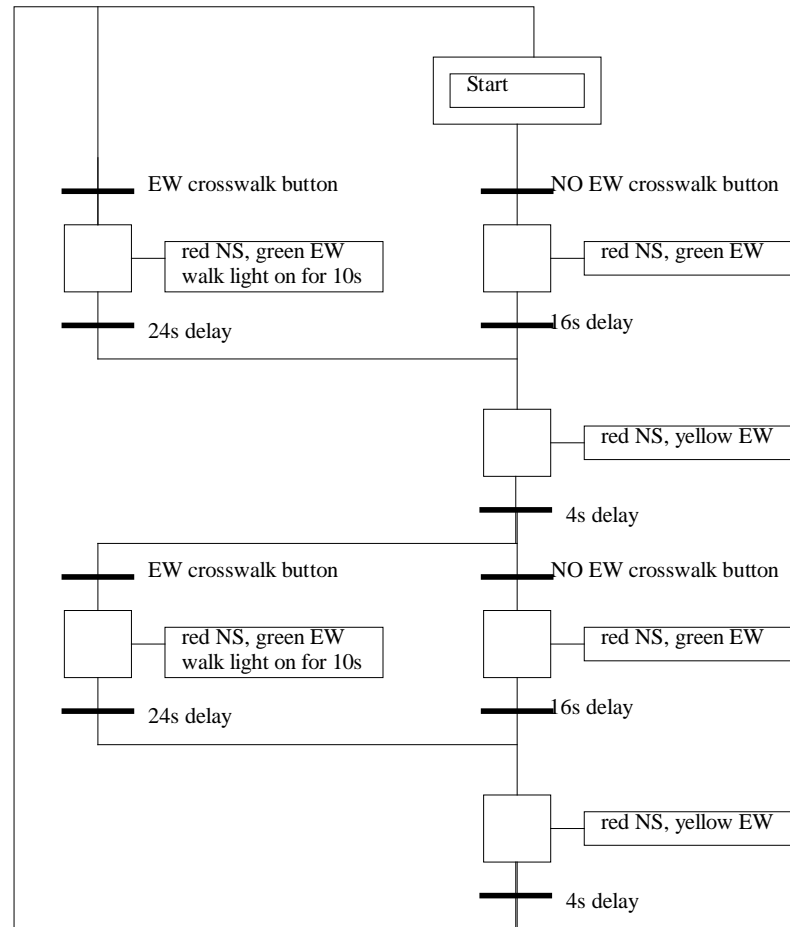
PUSH_STK:   LD    0
            ST    EMPTY    (* Pushed stack is not empty *)
            LD    PTR
            ADD   1
            ST    PTR
            EQ    NI        (* Overflow when PTR = NI *)
            ST    OFLO
            JMPC  ZRO_OUT
            LD    IN
            ST    STK[PTR]    (* Push IN onto STK *)
            JMP   SET_OUT

ZRO_OUT:    LD    0        (* OUT=0 for EMPTY or OFLO *)
SET_OUT:    ST    OUT
```

# IEC61131-3: Function Block Diagram



# IEC61131-3: Sequential Function Chart







Whimori

# How to make Street Fighters gentle

- New PLs and environments
  - abstract, problem-oriented development
  - simple and rigorous for safe programming
- Existing PLs + analysis and verification
  - clear understanding of semantics
  - tools for analysis



# Motivation

- A new development environment
  - based on IEC61131-3 standards
  - language processors based on formal semantics
  - supporting verification based on formal methods
  - supporting open data format
  - implemented on an open framework
- A testbed for verification of control programs
  - formally specified syntax and semantics
  - formally designed language processors

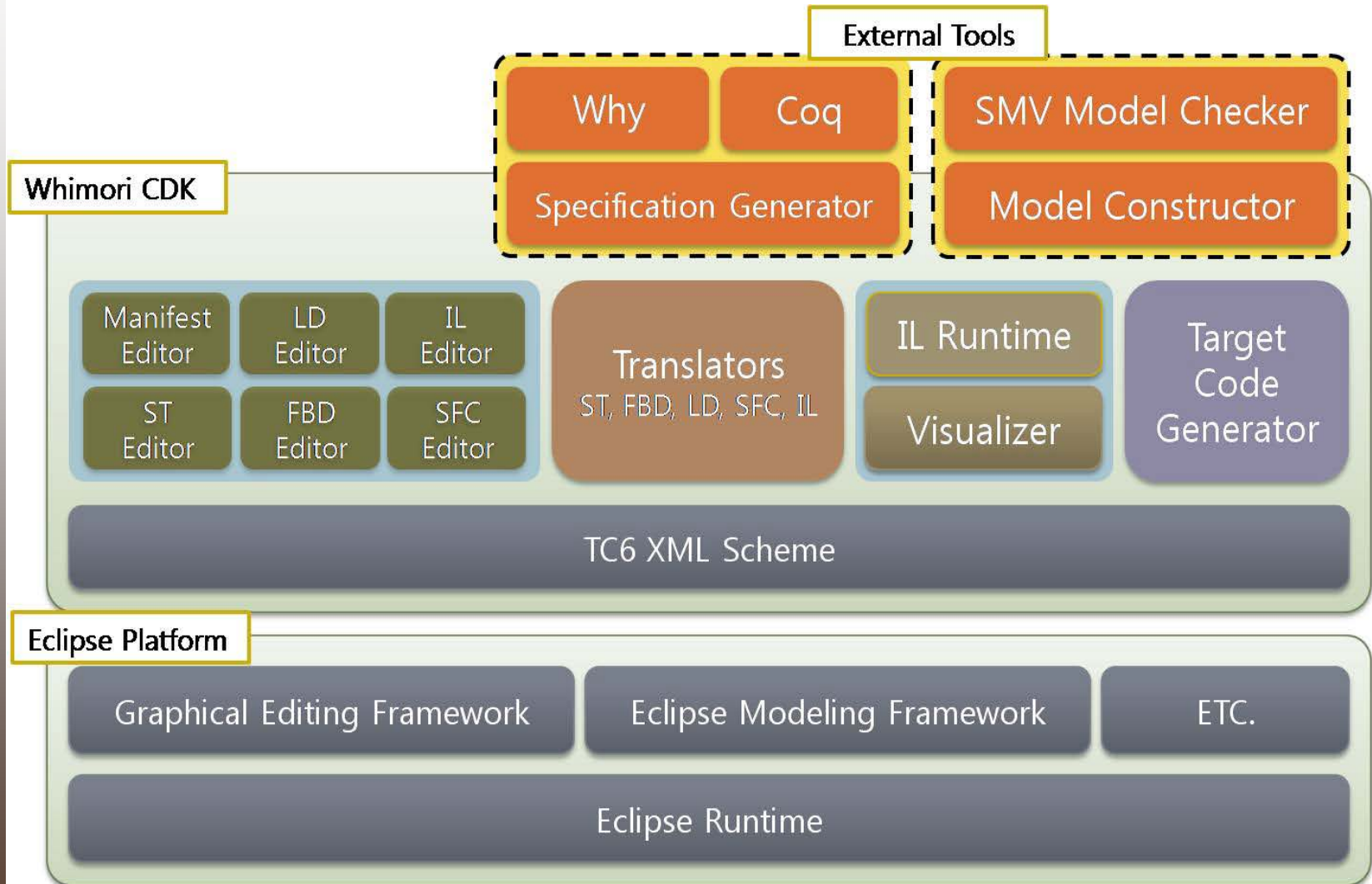


- IEC 61131-3 language standards
- Formal Semantics based
- Language Translators formally designed
- Virtual Machine formally designed
- TC6 XML Scheme as an open data format
- Eclipse platform as an open framework

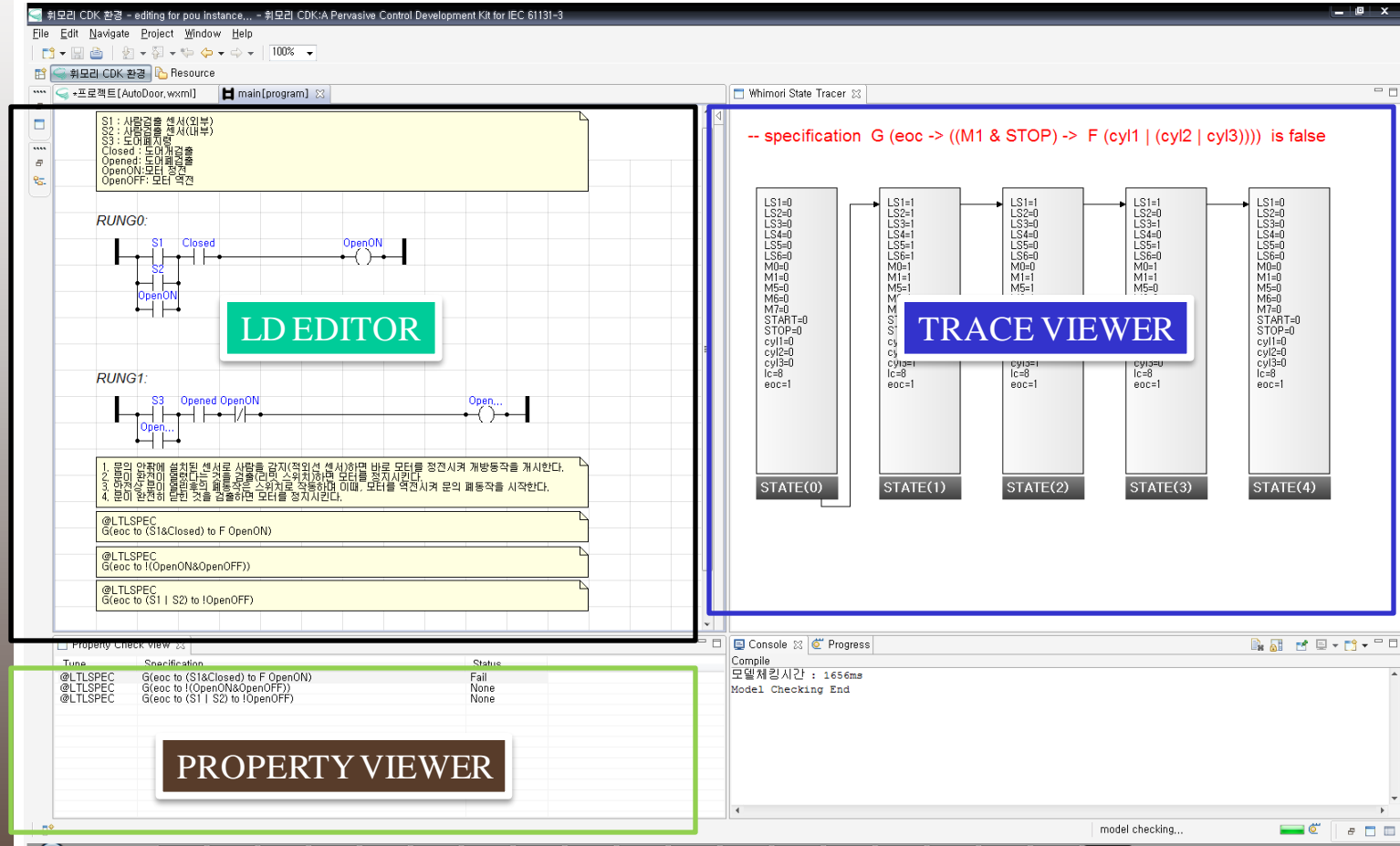
# PLC program verification

- which source language?
- having environment or not?
- scan cycle implicit or explicit?
- having timer or not?
- which model?
  - automata, transition systems, Petri net
  - logics, languages, calculi
  - constraints
- which verification engine?

# Whimori architecture



# Whimori + SMV model checker



The screenshot displays the Whimori software interface, which is used for developing and verifying PLC programs. It is divided into several key sections:

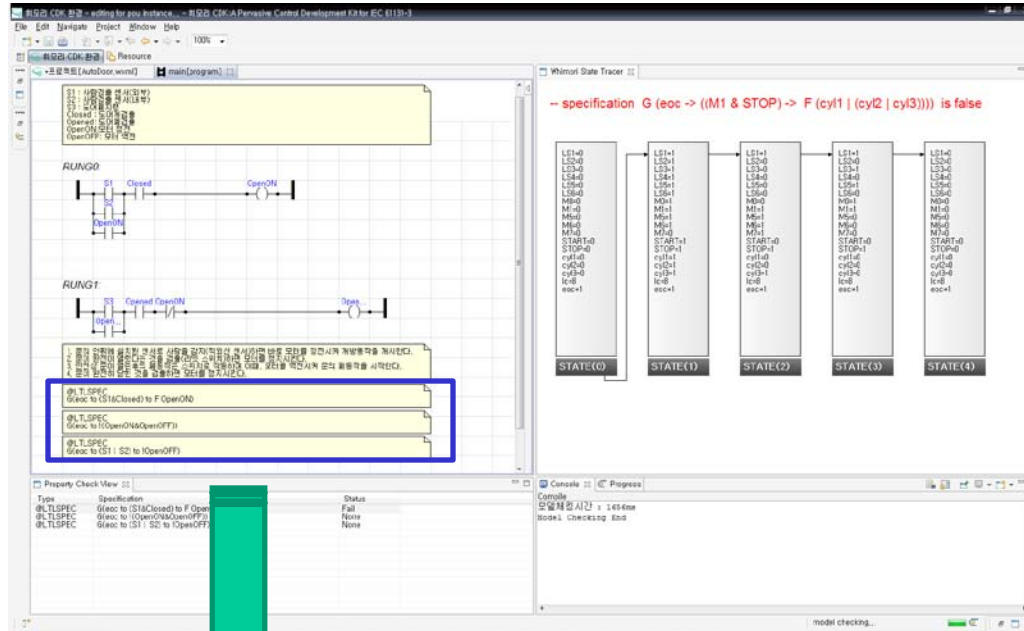
- LD EDITOR:** Shows two rungs of ladder logic. Rung 0 contains a normally open contact labeled 'S1' in series with a normally closed contact labeled 'S2', leading to a coil labeled 'OpenON'. Rung 1 contains a normally open contact labeled 'S3' in series with a normally open contact labeled 'OpenON', leading to a coil labeled 'Open...'. Below the rungs are three LTL specifications:
  - @LTLSPEC G(eoc to (S1&Closed) to F OpenON)
  - @LTLSPEC G(eoc to !(OpenON&OpenOFF))
  - @LTLSPEC G(eoc to (S1 | S2) to !OpenOFF)
- TRACE VIEWER:** Displays a state transition graph with five states labeled STATE(0) through STATE(4). Each state is represented by a vertical bar containing the values of various variables. A red text overlay at the top of this window states: "-- specification G (eoc -> ((M1 & STOP) -> F (cyl1 | cyl2 | cyl3))) is false".
- PROPERTY VIEWER:** A table showing the results of the model checking process.
 

Time	Specification	Status
@LTLSPEC	G(eoc to (S1&Closed) to F OpenON)	Fail
@LTLSPEC	G(eoc to !(OpenON&OpenOFF))	None
@LTLSPEC	G(eoc to (S1   S2) to !OpenOFF)	None
- Console:** Shows the compilation and model checking results:
 

```

Compile
모델채킹시간 : 1656ms
Model Checking End
      
```

# LTL Specification



The screenshot shows a software development environment with the following components:

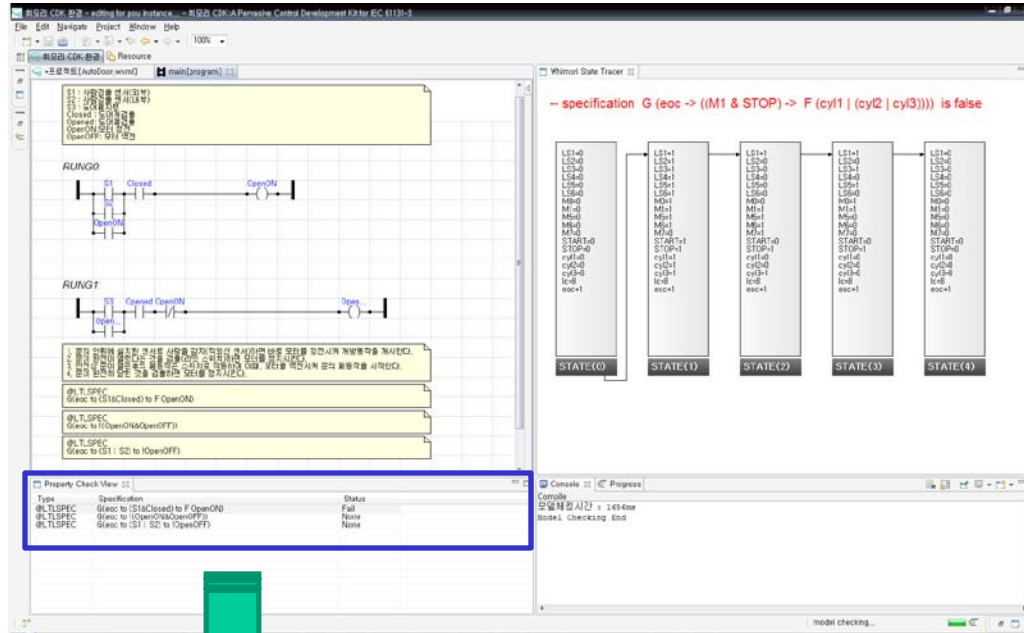
- Ladder Logic Diagram:** Shows two rungs, RUNG0 and RUNG1, with various logic elements like S1, S2, OpenON, and OpenOFF.
- LTL Specifications:** A list of specifications is shown, with three highlighted in a blue box:
  - @LTLSPEC G(eoc to (S1&Closed)) to F OpenON
  - @LTLSPEC G(eoc to !(OpenON&OpenOFF))
  - @LTLSPEC G(eoc to (S1 | S2) to !OpenOFF)
- Whimori State Tracer:** A state transition diagram showing five states (STATE(0) to STATE(4)) with transitions between them. A red text above it reads: "-- specification G (eoc -> ((M1 & STOP) -> F (cy1 | cy2 | cy3))) is false".
- Property Check View:** A table showing the status of the specifications:
 

Type	Specification	Status
@LTLSPEC	G(eoc to (S1&Closed)) to F OpenON	Fail
@LTLSPEC	G(eoc to !(OpenON&OpenOFF))	None
@LTLSPEC	G(eoc to (S1   S2) to !OpenOFF)	None
- Console:** Shows the execution time and the message "model checking done".



- @LTLSPEC  
G(eoc to (S1&Closed) to F OpenON)
- @LTLSPEC  
G(eoc to !(OpenON&OpenOFF))
- @LTLSPEC  
G(eoc to (S1 | S2) to !OpenOFF)

# Model Checking Results



The screenshot shows the Whimori State Tracer interface. On the left, there is a state transition diagram with two rungs, RUNG0 and RUNG1, and various states like S1, Closed, OpenON, and Stop. On the right, the Whimori State Tracer window displays a state transition diagram with five states labeled STATE(0) through STATE(4). The specification  $G(eoc \rightarrow ((M1 \& STOP) \rightarrow F(cyl1 | cyl2 | cyl3)))$  is shown as false. Below the diagram, a console window displays the following results:

Type	Specification	Status
@LTLSPEC	G(eoc to (S1&Closed) to F OpenON)	Fail
@LTLSPEC	G(eoc to !(OpenON&OpenOFF))	None
@LTLSPEC	G(eoc to (S1   S2) to !OpenOFF)	None



Type	Specification	Status
@LTLSPEC	G(eoc to (S1&Closed) to F OpenON)	Fail
@LTLSPEC	G(eoc to !(OpenON&OpenOFF))	None
@LTLSPEC	G(eoc to (S1   S2) to !OpenOFF)	None



# Counter Examples

Whimori CDK 환경 - editing for pou instance... - Whimori CDK: A Pervasive Control Development Kit for IEC 61131-3

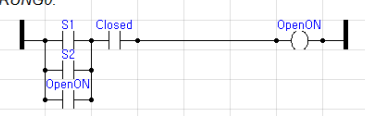
File Edit Navigate Project Window Help

Whimori CDK 환경 Resource

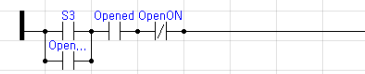
프로젝트 [AutoDoor.wxml] main[program]

S1 : 사람건 센서(외부)  
S2 : 사람건 센서(내부)  
S3 : 도어지령  
Closed : 도어 닫힘  
Opened : 도어 열림  
OpenON : 모터 정격  
OpenOFF : 모터 역전

RUNG0:



RUNG1:



1. S1, S2의 입력에 차등 센서 신호를 감지(외부 센서)하면 바로 모터를 정전시켜 개방 동작을 개시한다.  
2. S3의 입력에 차등 센서 신호를 감지(내부 센서)하면 바로 모터를 정전시켜 개방 동작을 개시한다.  
3. S1, S2의 입력에 차등 센서 신호를 감지(외부 센서)하면 바로 모터를 정전시켜 개방 동작을 개시한다.  
4. 이 단락을 실행하면 모터를 정전시켜 개방 동작을 개시한다.

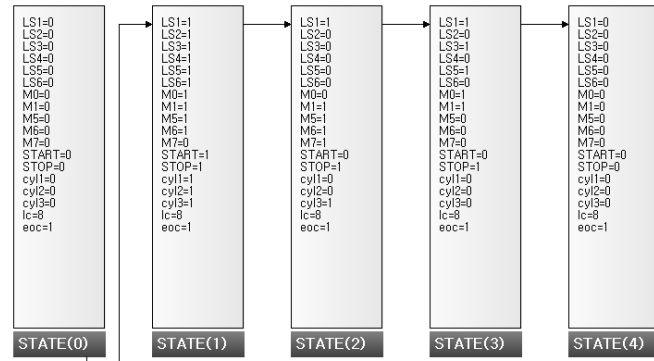
@LTLSPEC  
G(eoc to (S1&Closed) to F OpenON)

@LTLSPEC  
G(eoc to !(OpenON&OpenOFF))

@LTLSPEC  
G(eoc to (S1 | S2) to !OpenOFF)

Whimori State Tracer

-- specification G (eoc -> ((M1 & STOP) -> F (cyl1 | cyl2 | cyl3))) is false



STATE(0)

STATE(1)

STATE(2)

STATE(3)

STATE(4)

Property Check View

Type	Specification	Status
@LTLSPEC	G(eoc to (S1&Closed) to F OpenON)	Fail
@LTLSPEC	G(eoc to !(OpenON&OpenOFF))	None
@LTLSPEC	G(eoc to (S1   S2) to !OpenOFF)	None

Console Progress

Compile  
모델체크시간 : 1656ms  
Model Checking End

model checking...

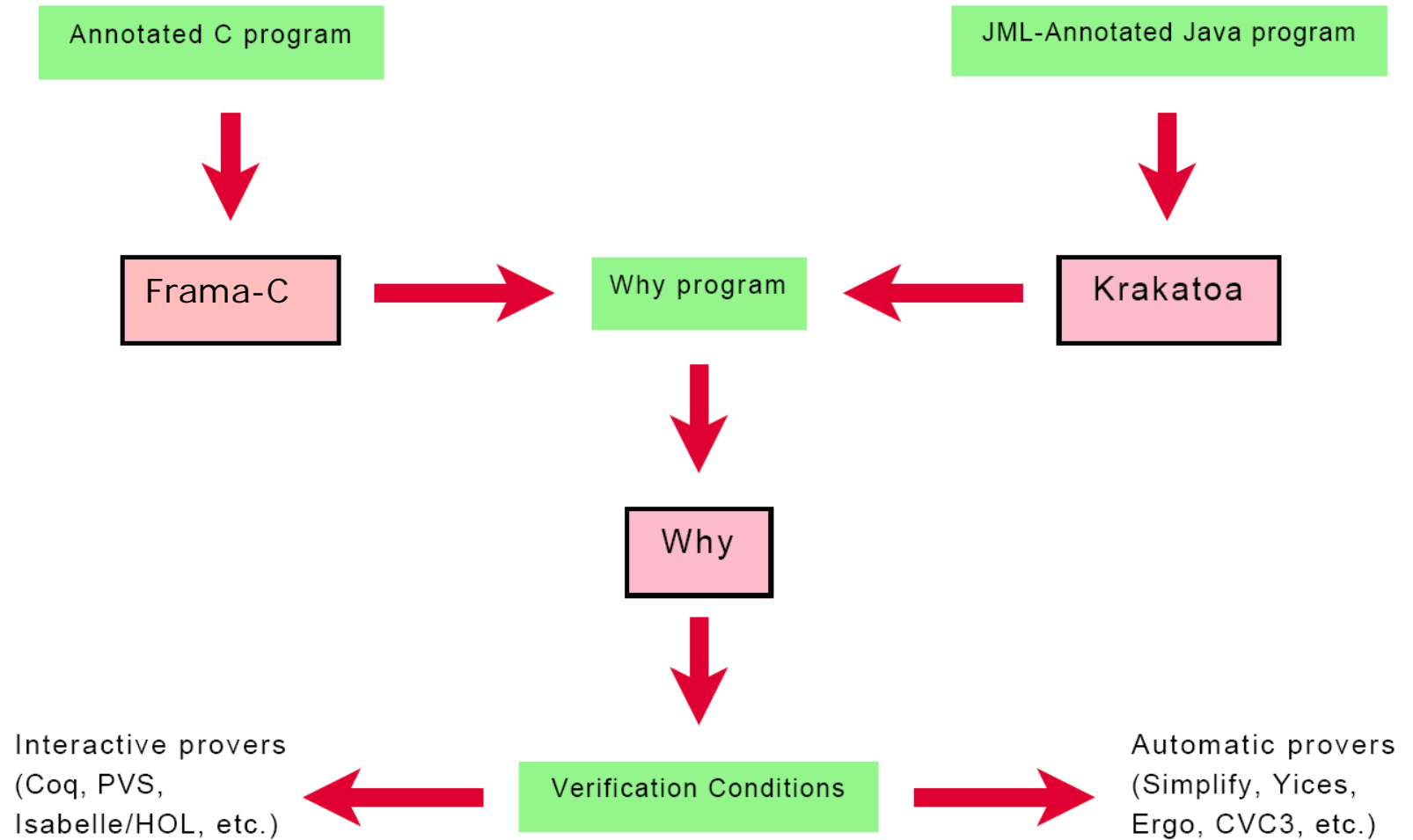
# Deductive Verification using Why

- Software Verification Platform
- verification condition generator(VCG)
- Back-end for verification of C/Java programs
- Why code = ML like program + annotations

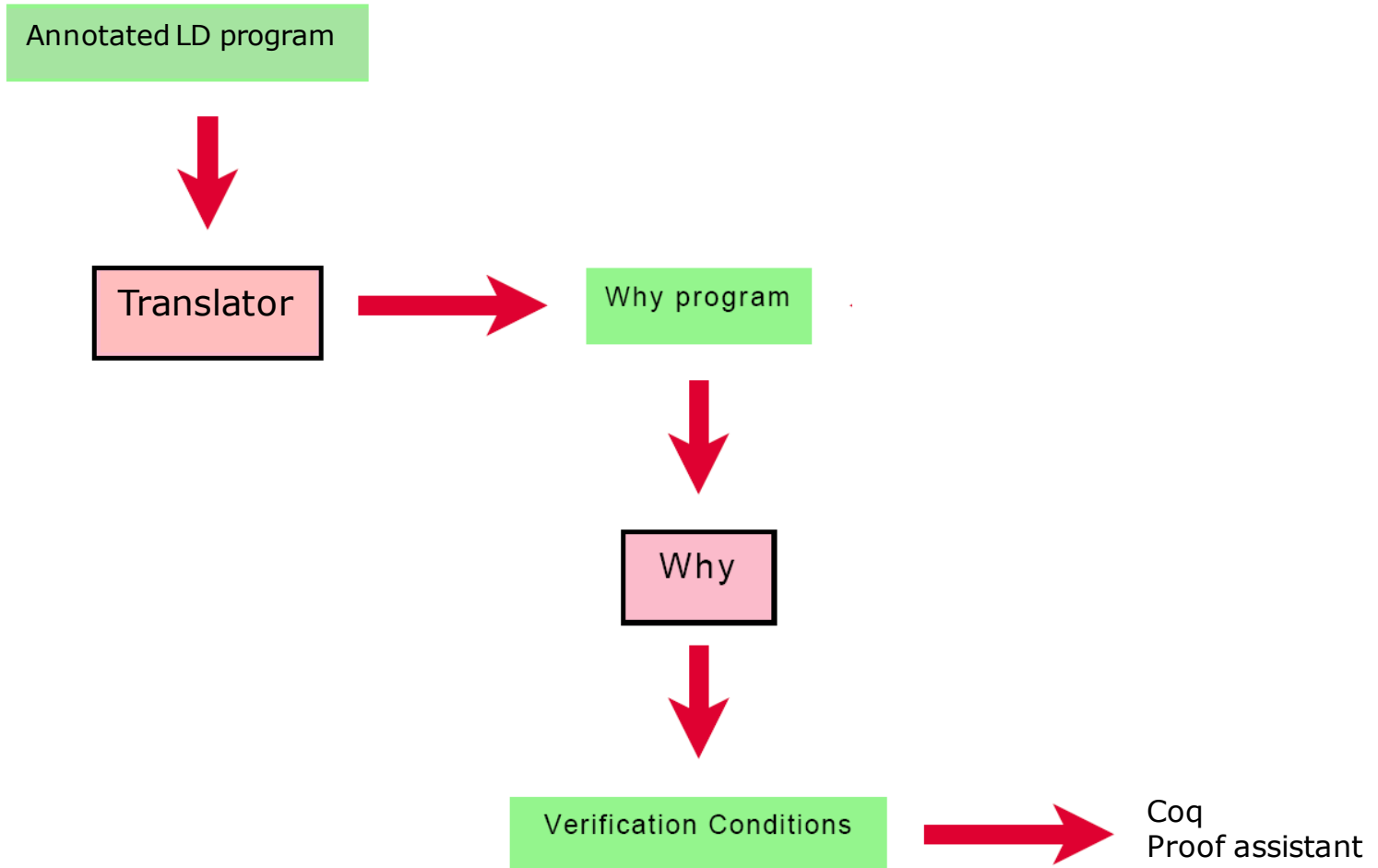


Whimori

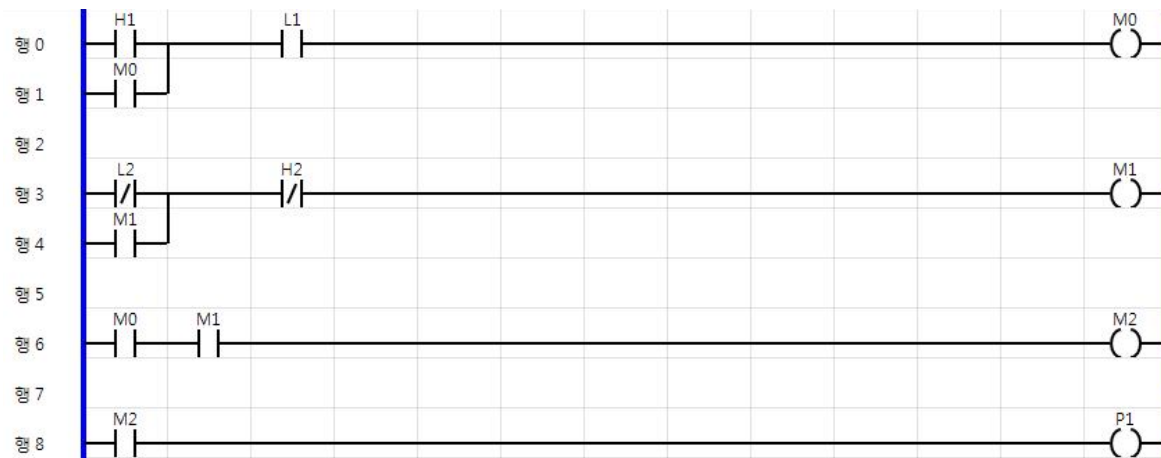
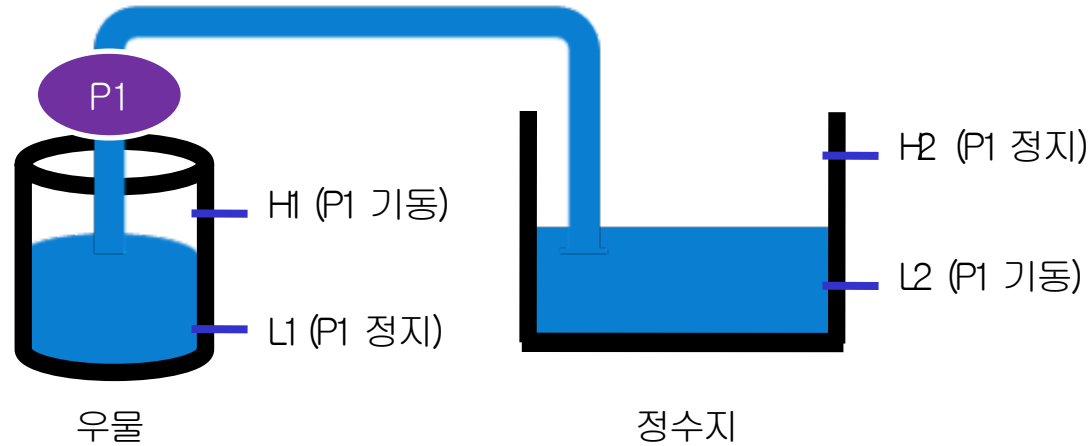
# Why?



# Deductive Verification of Ladder Diagram



# Very simple example



# Annotated Specification

```
/*@ ensures  
  @ ( ( !L1 || H2) => !P1 ) &&  
  @ ( (\old(M0) && \old(M1) && \old(M2) && \old(P1) ) =>  
  @ ( !H1 && L1 && !H2 ) =>  
  @ P1 )  
@*/
```

우물의 수위가 낮거나(L1 이하)  
정수지의 수위가 높을 때(H2 이상)  
과연 펌프는 동작하지 않는가?

**&&**

펌프가 동작하는 상태에서  
우물의 수위가 중간(H1과 L2 사이)이고  
정수지의 수위는 높지 않을 때(H2이하)  
계속 펌프는 동작하는가?



Whimori

# Why code

```
(*
 * l1, l2, h1, h2 : inputs
 * p1 : output
 *)
(* global variables *)
parameter p1 : int ref
(* the pump *)
parameter m0 : int ref
parameter m1 : int ref
parameter m2 : int ref
```

```
(* body *)
let water_impl =
  fun (l1 : int) (l2 : int) (h1 : int) (h2 : int) ->
    { }
    (
      begin
        (m0 := (if ( ((neq_int_ h1) (0)) || ((neq_int_ !m0) (0)))
                    && ((neq_int_ l1) (0)) )
              then (1)
              else (0)));
        (m1 := (if ( ((eq_int_ l2) (0)) || ((neq_int_ !m1) (0)))
                    && ((eq_int_ h2) (0)) )
              then (1)
              else (0)));
        (m2 := (if ((neq_int_ !m0) (0)) && ((neq_int_ !m1) (0)))
              then (1)
              else (0)));
        (p1 := !m2)
      end)
    { ((eq_int(l1, (0)) or neq_int(h2, (0))) -> not (neq_int(p1,
(0))))
      and ( ( neq_int(m0@, (0)) and neq_int(m1@, (0)) and
              neq_int(m2@, (0)) and neq_int(p1@, (0)) ) ->
            ( eq_int(h1, (0)) and neq_int(l1, (0)) and eq_int(h2,
(0)) ) ->
              (neq_int(p1, (0))) ) }
```



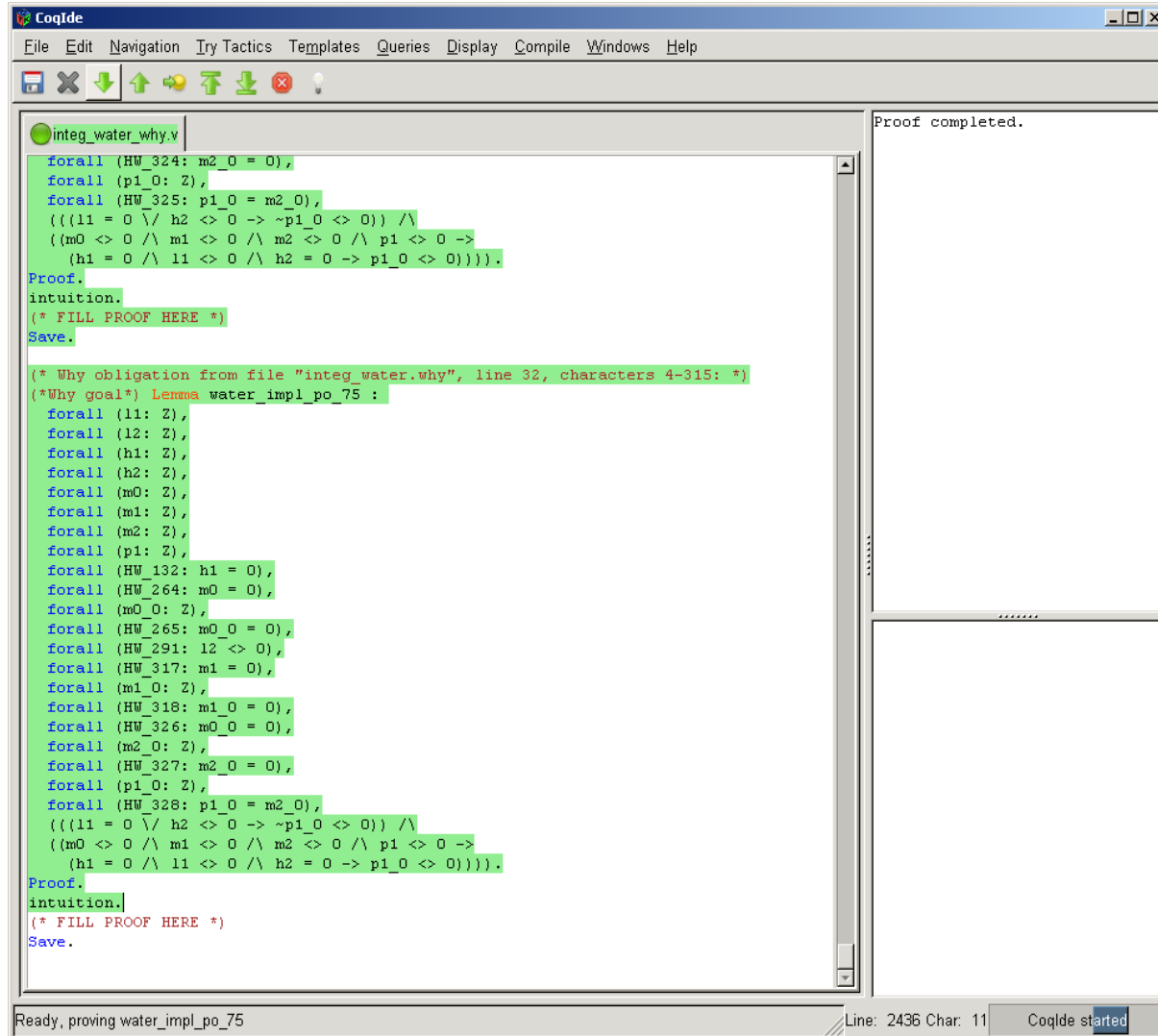
## Why(VCG)

- Generate Verification Conditions
- weakest preconditions calculus

$$\text{wp}(\{p'\} \text{ e } \{q'\}, q) = p' \wedge \forall \text{result}. \forall \omega. q' \Rightarrow q$$



# Proving Verification Conditions by Coq



```
CoqIde
File Edit Navigation Try Tactics Templates Queries Display Compile Windows Help

integ_water_why.v
forall (HW_324: m2_0 = 0),
forall (p1_0: Z),
forall (HW_325: p1_0 = m2_0),
(((l1 = 0 \\/ h2 <> 0 -> ~p1_0 <> 0) /\
(m0 <> 0 /\ m1 <> 0 /\ m2 <> 0 /\ p1 <> 0 ->
(h1 = 0 /\ l1 <> 0 /\ h2 = 0 -> p1_0 <> 0))).
Proof.
intuition.
(* FILL PROOF HERE *)
Save.

(* Why obligation from file "integ_water.why", line 32, characters 4-315: *)
(*Why goal*) Lemma water_impl_po_75 :
forall (l1: Z),
forall (l2: Z),
forall (h1: Z),
forall (h2: Z),
forall (m0: Z),
forall (m1: Z),
forall (m2: Z),
forall (p1: Z),
forall (HW_132: h1 = 0),
forall (HW_264: m0 = 0),
forall (m0_0: Z),
forall (HW_265: m0_0 = 0),
forall (HW_291: l2 <> 0),
forall (HW_317: m1 = 0),
forall (m1_0: Z),
forall (HW_318: m1_0 = 0),
forall (HW_326: m0_0 = 0),
forall (m2_0: Z),
forall (HW_327: m2_0 = 0),
forall (p1_0: Z),
forall (HW_328: p1_0 = m2_0),
(((l1 = 0 \\/ h2 <> 0 -> ~p1_0 <> 0) /\
(m0 <> 0 /\ m1 <> 0 /\ m2 <> 0 /\ p1 <> 0 ->
(h1 = 0 /\ l1 <> 0 /\ h2 = 0 -> p1_0 <> 0))).
Proof.
intuition.
(* FILL PROOF HERE *)
Save.

Proof completed.
```

Ready, proving water\_impl\_po\_75 Line: 2436 Char: 11 CoqIde started



**Whimori**

# Summary

- Whimori CDK: IDE for PLC programming
- based on Modern PL theory
- Verification tools featured
  - Model checking
  - Theorem proving