



## 2 Leo is a Subsystem of $\Omega_{\text{MEGA}}$

Leo has been realized as a part of the  $\Omega_{\text{MEGA}}$  framework. This framework (and thus also Leo) is implemented in CLOS [25], an object-oriented extension of Lisp.

**itself employ a Henkin**







OMEGA: show-commands leo-interactive  
[...]  
DECOMPOSE:           Applies



```

(implies (p (and (a m) (or (b m) (c m))))
  (p (and (a m) (dc-20735 c b))))))}
=====
Clause cl4 is #<Justified by ((CNF)) on (cl2)> :
cl4(1.5|1):{(+(p (and (a m) (or (b m) (c m)))))}
=====
Clause cl5 is #<Justified by
((RES 1 1) (RENAMING {(dc2 --> dc37)})) on (cl3 cl4)> :
cl5(1|2):{(-(= (p (and (a m) (dc37 c b)))
  (p (and (a m) (or (b m) (c m))))))}
=====
Clause cl7 is #<Justified by
((UNI {(?h113 --> [I am ?h122 ?h123.m])
  (?h107 --> [I am ?h120 ?h121.m])
  (?h101 --> [I am ?h111 ?h112. (?h111 (?h113 ?h111 ?h112))])
  (?h100 --> [I am ?h105 ?h106. (?h106 (?h107 ?h105 ?h106))])
  (dc37 --> [I am ?h98 ?h99. (or (?h100 ?h98 ?h99)
    (?h101 ?h98 ?h99))])})
  (RENAMING {}}) on (cl5)> :
cl7(0|2):{NIL}
=====
Clause cl14 is #<Justified by ((CNF)) on (cl7)> :
cl14(0|3):{NIL}
=====
===== clauses in proof: 7 =====

```

OMEGA:

**We now slightly modify our example problem and obtain a much harder one.**

$\exists$   $y$   $r d ( )$   $2( d ( )$   $u )$   $2 2 d ($   $2 2 d ( r d d ( )$   $2( d ( )$   $($   $1 1 11 1$   $1$   
 $n$   $(P P )( P )( P )( P ))P$   
 $n$

THEORY-LIST (SYMBOL-LIST) Theories whose definitions will be expanded: [()]  
Expanding the Definitions...  
[...]

OMEGA: show-clauses  
===== BEGIN =====  
The set of



## Leo's Architecture and Main Loop

Leo's basic architecture adapts the set of support approach. The four cornerstones of Leo's architecture (see Fig. 1) are:

⌈

⌋





**Step 13 (Integrate to SOS)** In the last step Leo integrates all pre-uni ed clauses in UNIFIED into the sorted store SOS. Forward and/or backward sub-  
sumption is employed depending on ending





```
Start proving ...
Loop: (100sec left)
#1 (SOS 2 USABLE 0 EXT-QUEUE 0 F0-LIKE 4)
(99sec left)
#2 (SOS 1 USABLE 1 EXT-QUEUE 0 F0-LIKE 4)
(98sec left)
[...]
(96sec left)
#9 (SOS 3 USABLE 4 EXT-QUEUE 0 F0-LIKE 12)
[...]
Calling bliksem process 22267 with time resource 50sec .
PARSING BLIKSEM OUTPUT ...
Bliksem has found a saturation.
[...]
(96sec left)
#10 (SOS 10 USABLE 5 EXT-QUEUE 0 F0-LIKE 20)
[...]
(94sec left)
#21 (SOS 39 USABLE 9 EXT-QUEUE 0 F0-LIKE 60)
Calling bliksem process 22454 with time resource 50sec .
bliksem Time Resource in seconds:
PARSING BLIKSEM OUTPUT ...
Bliksem has found a proof.
Bliksem's time:
; cpu time (non-gc) 0 msec user, 0 msec system
; cpu time (gc)    0 msec user, 0 msec system
; cpu time (total) 0 msec user, 0 msec system
; real time 174 msec
; space allocation:
; 416 cons cells, 0 symbols, 15,720 other bytes,
```

Leo has initially been implemented as a demonstrator system for extensional higher-order resolution in the context of the author's PhD thesis [4]. The experiments carried out with Leo so far, in

