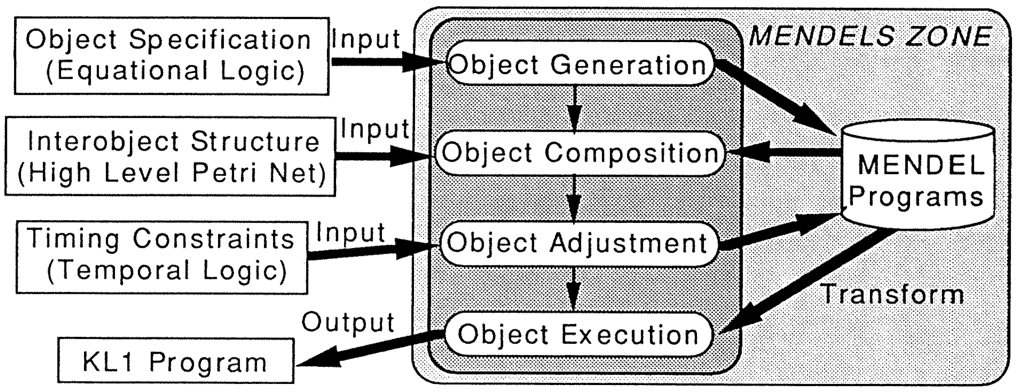
MENDELS ZONE
A Concurrent Program Development System

ABSTRACT

MENDELS ZONE is a software development system for concurrent programs. The system takes as input several kinds of specifications for MENDEL objects, verifies their correctness using automated theorem proving techniques, and generates correct KL1 programs semi-automatically.

KEY FEATURES

● **object generation:** Specifications, based on equational logic, for MENDEL objects are verified in parallel using term rewriting techniques and transformed into MENDEL objects automatically.
● **object composition:** MENDEL objects are represented by a high-level Petri net (MENDEL net). Users compose these objects with the aid of a graphical MENDEL net editor.
● **object adjustment:** Composed objects are automatically adjusted to satisfy supplied temporal logic specifications, which specify only timing constraints of composed objects. The object adjustment is carried out by parallel automated reasoning techniques with Petri nets and temporal logic.



SYSTEM CONFIGURATION

ABSTRACT

In the future, an increasing number of application programs will be distributed systems or concurrent systems. Debugging and testing of concurrent systems are very difficult, hence it is necessary to establish a methodology for the development of reliable programs. MENDELS ZONE is a system whose aim is the realization of such a methodology, and supports the development of correct concurrent programs through the combined use of several formal methods.

APPROACH

MENDEL is a concurrent programming language based on Petri net semantics, which can be compiled into KL1 and executed. A MENDEL program is composed of several objects which behave concurrently. MENDELS ZONE is a concurrent program development system for MENDEL, that adopts formal methods based on mathematics.

Formal methods are recognized as useful in developing reliable programs, but the following problems need to be solved before putting them into practical use:

1. It is difficult to describe a whole system with only one formal method.
2. Specifications become less comprehensible as their sizes grow larger.
3. Formal methods are not suitable for large scale systems.
4. The verification of specifications and the generation of programs are computationally costly.

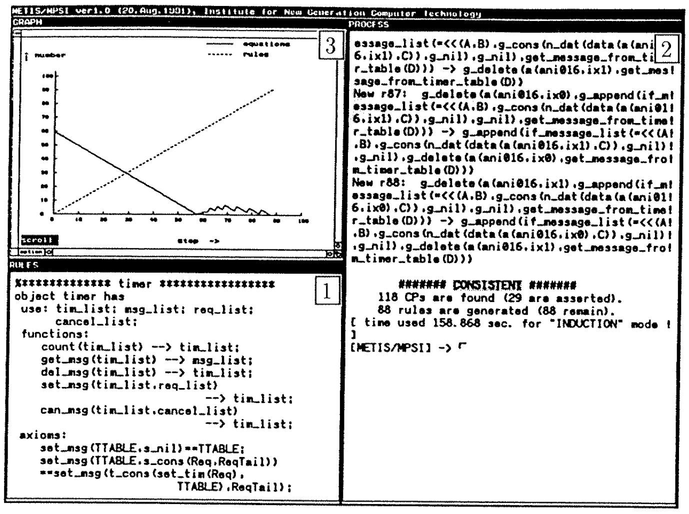
In MENDELS ZONE, the above problems are dealt with as follows. Objects behaviors are described by an algebraic specification based on equational logic, and timing constraints among objects are described using temporal logic. Visual specifications such as Petri nets are used to make formal specifications more understandable and lessen the burden of describing the specifications. Objects can be regarded as reusable parts because of their increased comprehensibility and modularity, which makes it possible to develop large scale systems. Theorem proving methods for both the equational logic and the temporal logic are used in order to verify specifications and transform them into programs. Parallel implementations of these methods contribute in shortening the time necessary for verification and transformation.

MENDELS ZONE has put formal methods into practical use as described above, and it becomes possible to verify the correctness of specifications and transform them into programs. As a result, reliable programs can be generated.

OUTLINE OF SYSTEM

The development of programs using MENDELS ZONE has two phases, namely, the "Algebraic Specification Phase" in which objects are generated and the "Petri Net Phase" in which objects are composed.

Figure 1 is a snapshot taken during the "Algebraic Specification Phase".


METIS/MPSI ver1.0 (20.Aug.1991), Institute for New Generation Computer Technology
GRAPH
PROCESS
RULES
3
2
1
number
equations
rules
step ->
scroll
option

```
********************* timer *********************
object timer has
 use: tim_list; msg_list; req_list;
      cancel_list;
 functions:
    count(tim_list) --> tim_list;
    get_msg(tim_list) --> msg_list;
    del_msg(tim_list) --> tim_list;
    set_msg(tim_list,req_list)
                              --> tim_list;
    can_msg(tim_list,cancel_list)
                              --> tim_list;
 axioms:
    set_msg(TTABLE,s_nil)==TTABLE;
    set_msg(TTABLE,s_cons(Req,ReqTail))
    ==set_msg(t_cons(set_tim(Req),
                         TTABLE),ReqTail);
```

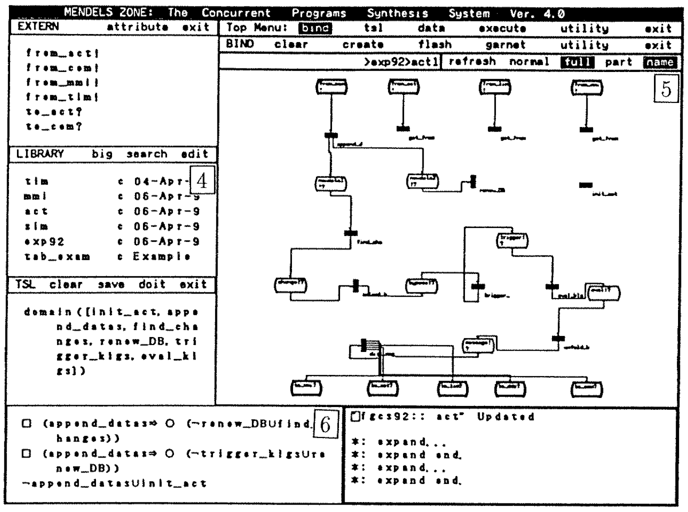
```
essage_list(=<<(A,B),g_cons(n_dat(data(a(ani
6,ix1),C)),g_nil),g_nil),get_message_from_ti
r_table(D))) -> g_delete(a(ani016,ix1),get_mes!
sage_from_timer_table(D))
New r87:  g_delete(a(ani016,ix0),g_append(if_m!
essage_list(=<<(A,B),g_cons(n_dat(data(a(ani01!
6,ix1),C)),g_nil),g_nil),get_message_from_time!
r_table(D))) -> g_append(if_message_list(=<<(A!
,B),g_cons(n_dat(data(a(ani016,ix1),C)),g_nil)!
,g_nil),g_delete(a(ani016,ix0),get_message_fro!
m_timer_table(D)))
New r88:  g_delete(a(ani016,ix1),g_append(if_m!
essage_list(=<<(A,B),g_cons(n_dat(data(a(ani01!
6,ix0),C)),g_nil),g_nil),get_message_from_time!
r_table(D))) -> g_append(if_message_list(=<<(A!
,B),g_cons(n_dat(data(a(ani016,ix0),C)),g_nil)!
,g_nil),g_delete(a(ani016,ix1),get_message_fro!
m_timer_table(D)))

        ####### CONSISTENT #######
     118 CPs are found (29 are asserted).
     88 rules are generated (88 remain).
[ time used 158.868 sec. for "INDUCTION" mode !
]
[METIS/MPSI] -> 
```

FIGURE 1

Users denote algebraic specifications for behavior (Method) of objects $\boxed{1}$. In order to describe specifications correctly, MENDELS ZONE provides several support functions: syntactical check, direct execution as a term rewriting system and verification with inductionless induction. When equations that the specifications should satisfy are given in $\boxed{2}$, the system verifies automatically whether they are true or not. Progress of the verification procedure can be observed in $\boxed{3}$. After verification, the specifications are transformed into MENDEL objects automatically.

Figure 2 is a snapshot taken during the "Petri Net Phase". Objects generated by automated transformation are registered in a Parts-Library $\boxed{4}$. Composition of objects is represented in terms of a high-level Petri net (MENDEL net), and is created using only mouse operations in the graphical MENDEL net editor $\boxed{5}$. In general, there are some timing constraints among methods in a object or among objects . In such a case, users denote the timing constraints using temporal logic $\boxed{6}$. MENDELS ZONE adjusts Petri nets automatically in order to satisfy the constraints. This program adjustment means synthesizing an arbiter that is attached to the original programs. A completed MENDEL program is compiled into a KL1 program, whose execution can be monitored within the environment provided by MENDELS ZONE.


MENDELS ZONE: The Concurrent Programs Synthesis System Ver. 4.0
EXTERN attribute exit
Top Menu: bind tsl data execute utility exit
BIND clear create flash garnet utility exit
>exp92>act1 refresh normal full part name
LIBRARY big search edit
TSL clear save doit exit
4
5
6

```
from_act!
from_com!
from_mmi!
from_tim!
to_act?
to_com?
```

```
tim          c 04-Apr-
mmi          c 06-Apr-9
act          c 06-Apr-9
sim          c 06-Apr-9
exp92        c 06-Apr-9
tab_exam     c Example
```

```
domain([init_act, appe
    nd_datas, find_cha
    nges, renew_DB, tri
    gger_kigs, eval_ki
    gs])
```

```
□ (append_datas⇒ ○ (¬renew_DBUfind_c
    hanges))
□ (append_datas⇒ ○ (¬trigger_kigsUre
    new_DB))
¬append_datasUinit_act
```

```
□fgcs92:: act" Updated

*: expand...
*: expand end.
*: expand...
*: expand end.
```

FIGURE 2

RESULT

We have developed an control system of a power station in order to examine the effectiveness of MENDELS ZONE for the development of a concurrent system. This control system was originally implemented directly in KL1, and we developed an almost identical system using MENDELS ZONE. Based on this experience, we feel confident in saying that a real system can be developed using MENDELS ZONE. A comparison between two development efforts reveals that MENDELS ZONE reduces debugging time although more time is needed for specifying and coding. This fact is an indication of effectiveness of verification based on formal methods.

OUTLINE OF DEMONSTRATION

We demonstrate the capabilities of MENDELS ZONE using the development example of the control system, and show the result of this development.