Ready-Simulation Is Not Ready to Express a Modular Refinement Relation

Françoise Bellegarde, Jacques Julliand, and Olga Kouchnarenko

LIFC, Univ. Franche-Comté, 16, route de Gray, 25030 Besançon Cedex France
{bellegar,julliand,kouchna}@lifc.univ-fcomte.fr,
http://lifc.univ-fcomte.fr

Abstract. The B method has been successfully used to specify many industrial applications by refinement. Previously, we proposed enriching the B event systems by formulating its dynamic properties in *LTL*. This enables us to combine model-checking with theorem-proving verification technologies. The model-checking of *LTL* formulae necessitates that the B event system semantics is a transition system. In this paper, we express the refinement relation by a relationship between transition systems. A result of our study shows that this relation is a special kind of simulation allowing us to exploit the partition of the reachable state space for a modular verification of *LTL* formulae. The results of the paper allow us to build a bridge between the above view of the refinement and the notions of observability characterized as simulation relations by Milner, van Glabbeek, Bloom and others. The refinement relation we define in the paper is a ready-simulation generalization which is similar to the refusal simulation of Ulidowsky. The way the relation is defined allows us to obtain a compositionality result w.r.t. parallel composition operation.

For complex systems, it is important in practice to associate a design by refinement with a design by a parallel composition of their components. This refinement relation has two main applications:
- it allows the splitting of the refined transition system into modules;
- it allows the construction of complex systems by a parallel composition of components.

It makes sense to qualify the refinement relation as being modular.

1 Introduction

In this paper, we express the refinement semantics as a relation between transition systems because we want to associate the verification of *LTL* formulae in the framework of a refinement design of reactive systems.

The B refinement method has been successfully used to specify many reactive systems: case studies such as an elevator [2], an industrial automatism [3], a steam-boiler case study [4], as well as industrial applications such as *MÉTÉOR* [7] by *Matra Transport International*, and the *SPECTRUM* project [23] by *GEC-Marconi Avionics Limited*.

T. Maibaum (Ed.): FASE 2000, LNCS 1783, pp. 266–283, 2000.

The B refinement method [1, 6] is used to specify reactive systems by event systems. In [12, 14], we propose to enrich this specification wiht dynamic properties formulated in the Linear Temporal Logic (LTL). We can then combine model-checking with theorem proving techniques. For that the verification has to take place both at the syntactic level for theorem proving and at the operational level for the verification of LTL formulae by model-checking. At the operational level, the specification is expressed with a transition system. At this level, the refinement relates transition systems between themselves. In [14, 18], we show that the refinement splits the refined transition system into modules. This allows us to verify some LTL properties separately on each module and to let what remains to be proved. For that, the refinement needs to be defined between transition systems. Therefore, the refinement verification takes place also at the operational level.

The results of the paper allow us to build a bridge between the above view of the refinement and the notions of observability characterized as simulation relations by Milner, van Glabbeek, Bloom and others.

We define the refinement relation as a simulation which allows us to exploit a partition of the refined reachable state space. With such a partition we are able to avoid the model-checking blow-up by verifying LTL formulae in a modular way [14]. Moreover, we want this refinement relation to be compositional w.r.t. parallel composition through refinement. That is why we call “modular” our refinement relation.

This paper is organized as follows. After giving preliminary notions in Section 2, Section 3 defines the behavior semantics of the transition systems derived from the B refinement design. Then, we define the transition system modular refinement relation in Section 4. Its expressiveness is studied in Sections 5 and 6. In Section 7, we illustrate the use of our framework on the example of the robot carrying parts from an arrival device towards two exit devices. Then, Section 8 explains how the refinement relation is used in the context of a verification tool set which combines automatic-proof and model-checking technologies. We end by some related works and some perspectives.

2 Preliminaries

In this paper we are concerned with a relationship between *transitions systems* which is a binary relation on their sets of states. In this framework, a predicate transformer is a function transforming sets of states into sets of states.

A **transition systems** is a pair $\langle S, \rightarrow \rangle$, where S is a set of states and $\rightarrow$ is a transition relation on S ($\rightarrow \subseteq S \times S$).

Definition 1. **(**$pre[\psi]$ **and** $post[\psi]$ **predicate transformers)** *Given a relation* ψ *between two set of states* S_1 *and* S_2 *(*$\psi : S_1 \times S_2$*), we define* $pre[\psi] : 2^{S_2} \times 2^{S_1}$ *and* $post[\psi] : 2^{S_1} \times 2^{S_2}$ *by*

- $pre[\psi] \overset{def}{=} \lambda X.\{q_1 \in S_1 \; s.t. \; (\exists q_2 \in X \; s.t. \; q_1 \psi q_2)\}$
- $post[\psi] \overset{def}{=} \lambda X.\{q_2 \in S_2 \; s.t. \; (\exists q_1 \in X \; s.t. \; q_1 \psi q_2)\}$

So, for $S'_2 \subseteq S_2$, $pre[\psi](S'_2)$ represents the set of predecessors of the states of S'_2 via the relation ψ, and, for $S'_1 \subseteq S_1$, $post[\psi](S'_1)$ represents the set of successors of the states of S'_1 via ψ. Some useful results concerning the *pre* and *post* predicate transformers can be found in [22] as, for example, the two following propositions.

Proposition 1. *For any relation ψ from a set S_1 to a set S_2 ($\psi \subseteq S_1 \times S_2$), we have:*

- *For any X_1, X_2 subsets of S_2, $pre[\psi](X_1 \cup X_2) = pre[\psi](X_1) \cup pre[\psi](X_2)$.*
- *For any X_1, X_2 subsets of S_1, $post[\psi](X_1 \cup X_2) = post[\psi](X_1) \cup post[\psi](X_2)$.*

We adopt the following notations:

- We denote by Id_S the *identity* function on 2^S.
- We denote by $\widetilde{\alpha}$ the dual of a function $\alpha : 2^{S_1} \to 2^{S_2}$ that is $\widetilde{\alpha} \overset{\text{def}}{=} \lambda X.\overline{\alpha(\overline{X})}$.
- We denote the composition of two relations $\psi \subseteq S_1 \times S_2$ and $\phi \subseteq S_2 \times S_3$ by their juxtaposition $\psi \; \phi$.
- We denote the composition of two predicate transformers $\alpha : X \to Y$ and $\beta : Y \to Z$ by $\beta \circ \alpha : X \to Z$.

Proposition 2. *Let be $\to \subseteq S \times S$. Then $pre[\to^2] = pre[\to] \circ pre[\to]$.*

We give hereafter the definition of Galois connections and some results about them. More information can, e.g., be found in [19, 21].

Definition 2. (Galois connections) *Let S_1 and S_2 be two sets of states. A connection from 2^{S_1} to 2^{S_2} is a pair of monotonic functions (α, γ), where $\alpha : 2^{S_1} \to 2^{S_2}$ and $\gamma : 2^{S_2} \to 2^{S_1}$, such that $Id_{S_1} \subseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \subseteq Id_{S_2}$.*

It is well-known that α and γ determine each other in a unique manner. These characterizations allow obtaining in [16] a proposition showing the links between the binary relation ψ from S_2 to S_1 and the connections from 2^{S_2} to 2^{S_1} in term of predicate transformers *pre* and *post*.

Proposition 3. (Connections generated by a binary relation on states) *If $\psi \subseteq S_1 \times S_2$, then the pair $(post[\psi], \widetilde{pre}[\psi])$ is a connection from 2^{S_1} to 2^{S_2}, and $(pre[\psi], \widetilde{post}[\psi])$ is a connection from 2^{S_2} to 2^{S_1}.*

3 Behavioral Semantics of Systems Derived from the B Design

In this paper we look at the B refinement at the operational semantic level since it splits the refined transition system into modules. This allows a modular verification of some LTL properties.

We define an operational semantics of a B specification under the form of a labeled transition system. We denote by S a set of states. We introduce a finite set of *variables* $V \stackrel{\text{def}}{=} \{x_1, \dots, x_n\}$. Let l be an injective function which allows us to give values to the states as a conjunction of variable/value equalities. Let q be a state, and $v_1, \dots, v_n$ be values, then $l(q)$ ($l(q)$ holds in q) is defined by $x_1 = v_1 \wedge \cdots \wedge x_n = v_n$. Usually, to describe LTL formulae semantics, the set of all the propositions holding in a state q is considered to be a label of q. We consider one of them $l(q)$ since, then, we know that a proposition P holds in q by $l(q) \Rightarrow P$ holds.

We call *invariant* a predicate I which holds on each state, and, as such, formulates a safety property of the system. A specification in B always requires such an invariant. A predicate I is an invariant of an interpreted transition system iff $\forall q \in S,\ l(q) \Rightarrow I$.

Let $Act \stackrel{\text{def}}{=} \{a, b, \dots\}$ be a nonempty alphabet of interpreted *actions.* The actions affect state variables. A labeled transition relation $\rightarrow$ ($\subseteq S \times Act \times S$) is defined as a set of triples (q, a, q') (written "$q \xrightarrow{a} q'$"). The interpreted transition system $TS = \langle S, Act, \rightarrow, l\rangle$ has the state space $S \stackrel{\text{def}}{=} \{q, q', q_1, \dots\}$, labeled transition relation $\rightarrow \subseteq S \times Act \times S$, and the state interpretation function l.

The transition relation $\rightarrow$ can be extended on a sequence of transitions in the standard way: q' is reachable from q, written $q' \in (\rightarrow)^*(q)$, (or there is a *path* σ from q to q') if there exist states $q_1, \dots, q_n$ and transitions $t_1, \dots, t_{n-1}$ respectively labeled by $a_1, \dots, a_{n-1}$ such that

$$(q =)\ q_1 \xrightarrow{a_1} q_2 \dots \xrightarrow{a_{n-1}} q_n\ (= q').$$

We note $\Upsilon_{TS}(q)$ the set of the paths of the transition system TS beginning in q. Given the path $\sigma = q \xrightarrow{a} q' \xrightarrow{b} q'' \xrightarrow{c} \dots$ in $\Upsilon_{TS}(q)$, we define its trace, note $tr(\sigma)$ by $tr(\sigma) \stackrel{\text{def}}{=} abc\dots$.

Moreover, given a state q, any proposition P such that $l(q) \Rightarrow P$ can be used to verify a LTL formula along a path which contains q in its trace.

As usual, the inverse relation $(\rightarrow)^{-1}$ denotes the predecessor relation on states, and we can say that q' is reachable from q, $q\ (\rightarrow)^*\ q'$, iff $q'\ \left((\rightarrow)^*\right)^{-1}\ q'$.

4 Modular Refinement Relation

In this section we consider two interpreted transition systems $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$ and $TS_2 = \langle S_2, Act_2, \rightarrow_2, l_2\rangle$ giving the operational semantics of two systems at two levels of refinement. We say that TS_2 is a *refinement* of TS_1.

The syntactical requirements of the B refinement are expressed as follows:

1. The refinement introduces *new* actions, so $Act_1 \subseteq Act_2$.
2. The invariant I_2, commonly called the *gluing* invariant (for instance, cf. [6]), expresses how the variables from the two interpreted transition systems are linked. More precisely, the invariant of the refined system is $I_2 \wedge I_1$.

4.1 Modular Refinement as State Space Partition

We exploit the B refinement so that we can build a partition of the state space.

First, we define a binary relation $\mu \subseteq S_2 \times S_1$ allowing us to express the relation between values of two consecutive interpreted transition systems.

Definition 3. (Glued states) *The state* $q_2 \in S_2$ *is* glued *to* $q_1 \in S_1$*, written as* $q_2 \; \mu \; q_1$*, iff* $l(q_2) \wedge I_2 \Rightarrow l(q_1)$.

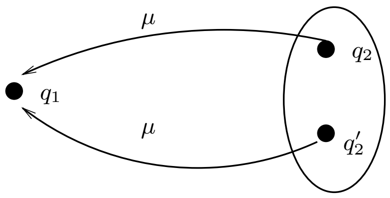

Fig. 1. State space partition idea

The glue relation μ allows us to define an equivalence relation $\sim_\mu$ between states of the second level transition system. Formally, two states q_2 and q_2' of the TS_2 are equivalent iff there exists a state q_1 of TS_1 s.t. $q_2 \; \mu \; q_1$ and $q_2' \; \mu \; q_1$ (see Figure 1). Indeed, we get an equivalence since giving two distinct states q_2 and q_2' we cannot have two distinct states q_1 and q_1' satisfying the above implication. When there are states of TS_2 glued to the state q_1 of TS_1, the state q_1 gives its name q_1 to an element of the partition.

Definition 4. (Equivalence class name) *Let* X *be an equivalence class of* $S_2/_{\sim_\mu}$*. The state* $q_1 \in S_1$ *is the name of* X *iff, for a state* $q_2 \in X$*, we have* $q_2 \; \mu \; q_1$.

4.2 Modular Refinement as a Relation

In this section we consider the refinement of an interpreted transition system as a simulation and we motivate it as a path set inclusion. Our purpose is to define a refinement relation to be a simulation allowing us to exploit the partition of the reachable state space for a modular verification of LTL formulae. For that, we restrict μ into a relation ρ which relates a refined transition system to one of its abstraction, and, we restrict ρ^{-1} into a relation ξ which relates a transition system to one of its refinement. These relations are useful to distinguish some elements of the partition of the state space.

This partition is used first, to prove an invariant of a module, and second, to verify propositional components of a LTL formula which is verified on a module.

From a Refined Transition System to One of Its Abstraction Let us call ρ a relation included into μ between the states of TS_2 and TS_1 which satisfies the following requirements:

1. In order to describe the refinement, we keep the transitions of TS_2 labeled over Act_1 but the *new* ones (from $Act_2 \setminus Act_1$) introduced by the refinement are considered as *non observable* τ moves. These τ moves hide the transitions of the modules viewed as interpreted transition systems. Indeed, the transitions of a module with state space S are the τ moves between the states of S (cf. Figure 2).

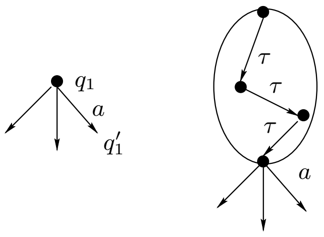

Fig. 2. Silent moves

2. In the above modules, it is certainly not desirable that τ moves take control forever. So, we want no deadlock and no infinite path of τ moves.

Let $Act_{1\tau} \stackrel{\text{def}}{=} Act_1 \cup \{\tau\}$. For each $a \in Act_{1\tau}$, we note $q \stackrel{a}{\Longrightarrow} q'$ when there is $n \geq 0$ such that $q \xrightarrow{\tau^n a} q'$. Intuitively, a transition $\stackrel{a}{\Longrightarrow}$ allows us to absorb a finite number of τ before the action a. Notice that Milner in [17] uses $\stackrel{a}{\Longrightarrow}$ for $\xrightarrow{\tau^n a \tau^m}$ with $n\,, m \geq 0$ but we force m to be equal to 0. The reason is that the occurrence of an action a determines the end of a path of a module. Notice that all the paths of a module are finite ones.

Definition 5. *Let $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$ and $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2, l_2\rangle$ be respectively a transition system and its refinement. Let a be in Act_1. The relation $\rho \subseteq S_2 \times S_1$ is defined as the greatest binary relation included into μ and satisfying the following clauses:*

1. **(strict transition refinement)** $(q_2\ \rho\ q_1 \wedge q_2 \xrightarrow{a}_2 q_2') \Rightarrow (\exists q_1'\ \ s.t.\ \ q_1 \xrightarrow{a}_1 q_1' \wedge q_2'\ \rho\ q_1')$
2. **(stuttering transition refinement)**
$(q_2\ \rho\ q_1 \wedge q_2 \xrightarrow{\tau}_2 q_2' \stackrel{a}{\Longrightarrow}_2 q_2'') \Rightarrow (\exists q_1'\ \ s.t.\ \ q_1 \xrightarrow{a}_1 q_1' \wedge q_2'\ \rho\ q_1 \wedge q_2''\ \rho\ q_1')$

$$\begin{tikzcd}
q_1 \arrow[d, "a"', "\exists"] & q_2 \arrow[l, "\rho"'] \arrow[d, "\tau"] \\
q_1' & q_2' \arrow[ul, "\rho"', "\exists"] \arrow[d, Rightarrow, "a"'] \\
 & q_2'' \arrow[ul, "\rho"', "\exists"]
\end{tikzcd}$$

3. **(lack of new deadlock)** $(q_2\ \rho\ q_1 \wedge q_2 \not\rightarrow_2) \Rightarrow (q_1 \not\rightarrow_1)$

4. **(non τ-divergence)** $q_2\ \rho\ q_1 \Rightarrow \neg\, (q_2 \xrightarrow{\tau} q_2' \xrightarrow{\tau} q_2'' \xrightarrow{\tau} \cdots \xrightarrow{\tau} \cdots)$

Notice that the presence of Clause 4 guarantees the monotonicity of an iterative construction of the relation ρ and, this way, the existence of this relation.

The relation ρ implements the modular refinement viewed as a path set inclusion. Given Υ_{TS_1} and Υ_{TS_2}, sets of paths of TS_1 and TS_2, we can see that

1. Clauses 1 and 2 of ρ-definition mean that every path of Υ_{TS_2} refines some path in Υ_{TS_1} (see Figure 3).

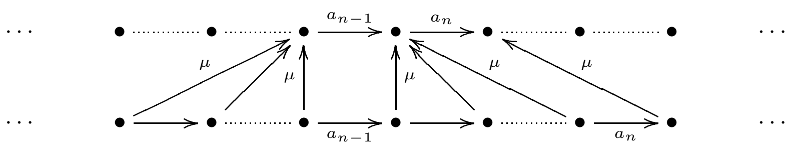

Fig. 3. Path refinement.

2. Clause 4 means that the refinement does not authorize infinite *new* paths composed only with new transitions in TS_2. In terms of graph, such infinite paths are cycles, so, there always must be a way out of these cycles, if any, by a strongly fair transition since such a transition, which is always eventually taken, forbids the infinite loop.
3. Clause 3 implies that any deadlock in Υ_{TS_2} corresponds to a deadlock in Υ_{TS_1} which means that new deadlocks are forbidden.

From an Abstract Transition System to One of Its Refinement

Definition 6. *Let $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$ and $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2, l_2\rangle$ be respectively a transition system and its refinement. Let a be in Act_1. The relation $\xi \subseteq S_1 \times S_2$ is the greatest binary relation included into ρ^{-1} and satisfying the following clause:*

(non-determinism) $(q_1 \xrightarrow{a}_1 q_1' \wedge q_1\ \xi\ q_2)) \Rightarrow (\exists\ q_2', q_2''\ s.t.\ q_1\ \xi\ q_2' \wedge q_2' \overset{a}{\Longrightarrow}_2 q_2'' \wedge q_1'\ \xi\ q_2'')$

$$\begin{tikzcd}
& q_1 \arrow[r, "\xi"] \arrow[dl, "a"'] \arrow[d, "a"'] \arrow[dr, "\xi", "\exists"'] & q_2 \\
\cdot & q_1' \arrow[dr, "\xi", "\exists"'] & q_2' \arrow[d, Rightarrow, "a", "\exists"'] \\
& & q_2''
\end{tikzcd}$$

The construction stays monotonic.

Again, we explain the ξ relation in terms of the path set inclusion. If there is an *internal* non-deterministic choice among the transitions of Υ_{TS_1} which begins at the same state q_1 (see Definition 6), then there exists (at least) one of these transitions which is refined by some path $\sigma_2 \in \Upsilon_{TS_2}$ (see Figure 4).

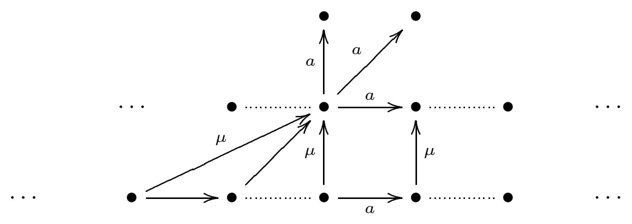

Fig. 4. Internal non-deterministic choice refinement.

Intuitively, the above notions of refinement mean that we observe the system more often taking into account additional details about its behavior. In terms of traces, the trace of the path σ_1 is embedded into the trace of the transitions in σ_2, or, in other words, we find the path σ_1 by removing the transitions labeled by *new* (from $Act_2 \setminus Act_1$) actions from the path σ_2 but this holds only if the transition systems are deterministic. In the presence of a non-deterministic choice, it is possible that some traces disappear among a non-deterministic set of transitions, but at least one trace remains.

The relation ξ allows us to define some elements of a partition of the refined state space $S_2/_{\sim_\mu}$ as modules, i.e., $X \in S_2/_{\sim_\mu}$ of name $q_1 \in S_1$ (i.e. the equivalence class name, see Definition 4) is also a module (named q_1) if and only if, for all the states q_2 in X, we have $q_1 \xi q_2$. So, from a designer point of view, "ξ holds" means that TS_1 is refined by TS_2.

As a consequence of the definition of ξ, there can be no deadlock and no livelock inside a module. Moreover, each path $\tau^* a$ refines a a-transition beginning in q_1. Finally, the set of the labels of all the transitions beginning in q_1 is equal to the set of the labels a of all the paths $\tau^* a$. This last consequence will be shown in Section 5.2.

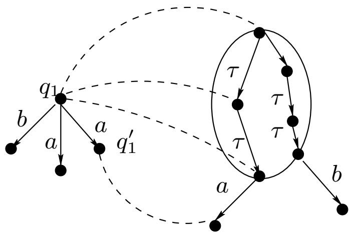

Fig. 5. Module

5 Modular Refinement as a Simulation

In this section, we consider the modular refinement relation $\eta \overset{\text{def}}{=} \xi^{-1}$. So, from a designer point of view, η holds means that TS_2 refines TS_1. We show that the modular refinement relation can be viewed as a special kind of τ-*simulation* which is derived from the observational equivalence of Milner [17] between a refined transition system and its abstraction. Finally, it is a generalization of the *ready* simulation of [10].

5.1 Modular Refinement as a τ-Simulation

In this section, we are interested in a view of the modular refinement as a τ-simulation from TS_2 to TS_1.

From the relation η, Proposition 3 allows us to generate a Galois connection $(post[\eta], \widetilde{pre}[\eta])$. The construction is inspired by [16]. Notice that the above authors are looking for building an abstraction when we are interested in a modular refinement. We choose this approach to show a τ-simulation because we see better the role of the modules by the way of the predicate transformers $post[\eta]$ and $\widetilde{pre}[\eta]$. Notice that the Milner's approach observes each time two related states, so it does not allow to consider a whole module (a set of states) which is related to one abstract state.

Let X be a subset of the refined state space S_2 $(X \subseteq S_2)$. Then $post[\eta](X)$ is the set of the module names, i.e.,

$$post[\eta] \overset{\text{def}}{=} \lambda X.\{q_1 \in S_1 s.t.\ (\exists q_2 \in X\ s.t.\ q_2\ \eta\ q_1)\}$$

Let X be a subset of the abstract state space S_1 $(X \subseteq S_1)$. Then $\widetilde{pre}[\eta](X)$ is the union of the state space of the modules named in X, i.e.,

$$\begin{aligned}\widetilde{pre}[\eta] \overset{\text{def}}{=} \lambda X. \textstyle\bigcup_{i \in I} X_i\ s.t.& \\ &(\forall i.\ i \in I \Rightarrow X_i \in S_2/_{\sim_\mu}) \wedge (\forall q_2.q_2 \in X_i \Rightarrow (\exists q_1 \in X\ s.t.\ q_2 \eta q_1))\end{aligned}$$

By using this Galois connection we show that the modular refinement is a kind of τ-simulation which is derived from the observational equivalence of Milner [17]. Now we give the definition of a η-simulation.

Definition 7. **($\sqsubseteq_\eta$)** *Let $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2\rangle$ and $TS_1 = \langle S_1, Act_1, \rightarrow_1\rangle$ be two transition systems and η be a relation from S_2 to S_1 ($\eta \subseteq S_2 \times S_1$). Define $S_2 \sqsubseteq_\eta S_1$ if and only if $((\xrightarrow{\tau}_2)^*)^{-1}(\xrightarrow{a}_2)^{-1}\ \eta\ \subseteq\ \eta\ (\xrightarrow{a}_1)^{-1}$ for all $a \in Act_1$.*

If $S_2 \sqsubseteq_\eta S_1$, we say that S_2 η-simulates S_1. This η-simulation is closely related to the τ-simulation in the sense of Milner. The difference is that in the *tau*-simulation the action a can be the empty action, and that τ-transitions can follow a.

In Theorem 1, we show that, by modular refinement, S_2 η-simulates S_1. For that, we need another predicate transformer in the refined transition system. Recall that a refined transition system TS_2 has silent moves whereas an abstract transition system TS_1 has no silent moves. For TS_2, the predicate transformer $pre[\rightarrow_2]$ is extended to $pre^{\tau^* a}[\rightarrow_2]$ to take into account the silent moves.

Definition 8. **($pre^{\tau^* a}[\rightarrow_2]$ predicate transformer)** *Given the relation $\rightarrow_2$: $S_2 \times S_2$, we define $pre^{\tau^* a}[\rightarrow_2]$: $S_2 \times S_2$ by*
$pre^{\tau^* a}[\rightarrow_2] \overset{def}{=} \lambda X_2.\{q_2 \in S_2$ s.t. $(\exists q_2' \in X_2$ s.t. $q_2' \overset{a}{\Longrightarrow} q_2)\}$.

The definition of $pre^{\tau^* a}$ is not a problem because there is no infinite τ-path (see Clause 4 of Definition 5). Notice that Proposition 1 applies to $pre^{\tau^* a}[\rightarrow_2]$ by finite composition of $pre[\rightarrow_2]$ because of Proposition 2.

We need the following lemma to prove that S_2 η-simulates S_1.

Lemma 1. *Let η be the relation between the two transition systems $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2, l_2\rangle$ and $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$. Let (α, γ) be the Galois connection $(post[\eta], \widetilde{pre}[\eta])$. Then, we have*

$$\alpha\ \circ\ pre^{\tau^* a}[\rightarrow_2]\ \circ\ \gamma\ \subseteq\ pre[\rightarrow_1]\ .$$

Proof. Let q_i be the name of a module Y_i (see Definition 4). Then

$$\alpha\ \circ\ pre^{\tau^* a}[\rightarrow_2](Y_i)$$

gives the modules names of the set $P = pre^{\tau^* a}[\rightarrow_2](Y_i)$ by Definition of α. All the elements of the set P are related to a predecessor of q_i by Clauses 1 and 2 of Definition 5. Therefore, this is included into the set $pre[\rightarrow_1](\{q_i\})$.

Let X be a subset of the abstract state space S_1 ($X \subseteq S_1$). Then, $\gamma(X) = \bigcup_{i\in I} Y_i$ where the Y_i's are the modules in S_2 the names of which are elements $\{q_1, q_2, \dots, q_i, \dots\}$ of X. Therefore,

$$\alpha\ \circ\ pre^{\tau^* a}[\rightarrow_2]\ \circ\ \gamma(X) = \alpha\ \circ\ pre^{\tau^* a}[\rightarrow_2]\ \big(\bigcup_{i\in I} Y_i\big)$$

which is equal to $\bigcup_{i\in I}\big(\alpha\ \circ\ pre^{\tau^* a}[\rightarrow_2]\ (Y_i)\big)$ by Propositions 1 and 2.

Since $\alpha\ \circ\ pre^{\tau^* a}[\rightarrow_2]\ (Y_i)$ is included into $pre[\rightarrow_1](\{q_i\})$ for all $i \in I$, we obtain that $\bigcup_{i\in I}\big(\alpha \circ pre^{\tau^* a}[\rightarrow_2]\ (Y_i)\big)$ is included into $\bigcup_{i\in I} pre[\rightarrow_1](\{q_i\})$. The latter is included into $pre[\rightarrow_1](X)$.

Theorem 1. *Let η be the relation between the two transition systems $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2, l_2\rangle$ and $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$. Then $S_2 \sqsubseteq_\eta S_1$.*

We slightly adapt the proof from [16] to our case by using Lemma 1, Definition 2 and Definition 7.

So, the modular refinement relation is an η-simulation. However, this result uses only Definition 5 of ρ. In the next section, we will point up the role played by the non-determinism.

5.2 Modular Refinement as a Generalization of the Ready-Simulation

In this section, we see the modular refinement relation η as a generalization of the ready-simulation of [10].

For that we will keep the ready-set definition (see [9, 10]) for the abstract system as $readies(q_1) = \{a \in Act_1 \ s.t. \ q_1 \xrightarrow{a}_1\}$, and, we will simply replace $\rightarrow_2$ by $\Longrightarrow_2$ for the refined system, i.e., $readies(q_2) = \{a \in Act_{1_\tau} \ s.t. \ q_2 \overset{a}{\Longrightarrow}_2\}$. Moreover, we take into account the non-divergence of τ.

Theorem 2. *Let $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$ and $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2, l_2\rangle$ be two transition systems. If $(q_2 \ \eta \ q_1)$ then $readies(q_1) = \bigcup_{q_2 \ s.t. \ q_2 \eta q_1} readies(q_2)$.*

Proof. $\subseteq$) It is immediate by Definition 6.
$\supseteq$) It is immediate by Clauses 1 and 2 of Definition 5.

Therefore, η implies the equality of the *readies* as it is the case for the ready simulation of [10]. This result points up that the set of a module exiting actions is equal to the set of this module name (see Definition 4) exiting actions.

Furthermore, the equality of the *readies* allows us to define ξ as follows:

Definition 9. *Let $TS_1 = \langle S_1, Act_1, \rightarrow_1, l_1\rangle$ and $TS_2 = \langle S_2, Act_{1\tau}, \rightarrow_2, l_2\rangle$ be respectively a transition system and its refinement. Let a be in Act_1. The relation $\xi \subseteq S_1 \times S_2$ is the greatest binary relation included into ρ^{-1} and satisfying the following clause:*
(readies) $(q_1 \xrightarrow{a}_1 q_1' \wedge q_1 \ \xi \ q_2)) \Rightarrow (readies(q_1) = \bigcup_{q_2 \ s.t. \ q_1 \xi q_2} readies(q_2))$

Theorem 3. *Definition 6 and Definition 9 are equivalent.*

Proof. $\Rightarrow$) It is immediate by Theorem 2.
$\Leftarrow$) It is immediate because Clause readies of Definition 9 implies Clause non-determinism of Definition 6.

Notice that Definition 9 is easily implementable since it is enough to verify the equality of the *readies* when verifying that Definition 5 holds.

6 Compositionality of the Modular Refinement

We give a result concerning the compositionality of the modular refinement relation η w.r.t. a parallel composition operation which is important for the application of the modular refinement in practice since it allows to build complex systems by a parallel composition of components.

Definition 10. (Parallel composition) *Let $TS_i = \langle S_i, Act_i, \rightarrow_i, l_i\rangle$ and $TS_j = \langle S_j, Act_j, \rightarrow_j, l_j\rangle$ be two transition systems. The parallel composition of TS_i and TS_j is $TS_i \| TS_j \overset{\text{def}}{=} \langle S, Act_\tau, \rightarrow, l\rangle$ where S is the set of the $q\|q''$ ($q \in S_i$ and $q'' \in S_j$), Act is the union of Act_i and Act_j, l is the product of l_i and l_j. Let $\alpha \in Act_\tau$, for $k \in \{i,j\}$, the transition relation $\rightarrow$ is defined by combining individual actions in parallel as follows.*

$$\text{[PAR1]}\ \frac{q \overset{\alpha}{\rightarrow}_k q'}{q\|q'' \overset{\alpha}{\rightarrow} q'\|q''} \qquad \text{[PAR2]}\ \frac{q \overset{\alpha}{\rightarrow}_k q'}{q''\|q \overset{\alpha}{\rightarrow} q''\|q'}$$

This definition means that all moves of parallel composition are moves of either TS_1 or of TS_2.

In the following theorem, η denotes the modular refinement relation linking transition systems as well as their parallel compositions.

Theorem 4. *Let $q_2\ \eta\ q_1$ and $q_4\ \eta\ q_3$. We have:*

1. $q_2\|q_4\ \eta\ q_1\|q_3$
2. $q_4\|q_2\ \eta\ q_3\|q_1$

Proof. We prove the result for 1, the second proof being similar.

For the result 1, we show that $\mathcal{S}$ verifies clauses of Definitions 5 and 6, where

$$\mathcal{S} \overset{\text{def}}{=} \{(q_1\|q_3, q_2\|q_4)\ s.t.\ q_2\ \eta\ q_1\ \&\ q_4\ \eta\ q_3\}$$

for $q_1, q_2, q_3, q_4 \in S$. Now, suppose $(q_1\|q_3, q_2\|q_4) \in \mathcal{S}$.

1. First, we consider Definition 5. Let $q_2\|q_4 \overset{\alpha}{\rightarrow} \tilde{q}$ with $\alpha \in Act_\tau$. There are four cases:
   (a) $q_2 \overset{a}{\rightarrow}_2 q_2'$, and $\tilde{q} = q_2'\|q_4$. Then, because $q_2\ \eta\ q_1$, we have $q_1 \overset{a}{\rightarrow}_1 q_1'$ with $q_2'\ \eta\ q_1'$ by Clause 1 of Definition 5; hence also $q_1\|q_3 \overset{a}{\rightarrow} q_1'\|q_3$ by $[PAR1]$ rule of Definition 10, and $(q_1'\|q_3, q_2'\|q_4) \in \mathcal{S}$.
   (b) $q_4 \overset{a}{\rightarrow}_4 q_4'$, and $\tilde{q} = q_2\|q_4'$. We get the result in the same way than the case above by using $[PAR2]$ rule of Definition 10.
   (c) $\alpha = \tau$, and $q_2 \overset{\tau}{\rightarrow}_2 q_2'$. Moreover, we derive $q_2' \overset{a}{\Longrightarrow}_2 q_2''$ and $q_2\|q_4 \overset{\tau}{\rightarrow} q_2'\|q_4 \overset{a}{\Longrightarrow} q_2''\|q_4$ $(= \tilde{q})$. Then, because $q_2\ \eta\ q_1$, we have $q_1 \overset{a}{\rightarrow}_1 q_1'$ with $q_2'\ \eta\ q_1$ and $q_2''\ \eta\ q_1'$ by Clause 2 of Definition 5; hence, we also get $q_1\|q_3 \overset{a}{\rightarrow} q_1'\|q_3$ by $[PAR1]$ rule of Definition 10, and $(q_1\|q_3, q_2'\|q_4)$ and $(q_1'\|q_3, q_2''\|q_4)$ are in $\mathcal{S}$.

   (d) $\alpha = \tau$, and $q_4 \xrightarrow{\tau}_4 q_4'$. We have the result by the same argument than the case above.
2. Second, we consider Definition 6. Let $q_1 \| q_3 \xrightarrow{\alpha} \tilde{q}$ with $\alpha \in \mathit{Act}$. There are two cases:
   (a) $q_1 \xrightarrow{a}_1 q_1'$, and $\tilde{q} = q_1' \| q_3$. Then, because $q_2 \ \eta \ q_1$, we have $q_1 \ \xi \ q_2$. By Definition 6, there exists q_2', q_2'' $\texttt{s.t.}$ $q_1 \ \xi \ q_2' \wedge q_2' \overset{a}{\Longrightarrow}_2 q_2'' \wedge q_1' \ \xi \ q_2''$. By $[PAR1]$ rule of Definition 10, we have $q_2' \| q_4, q_2'' \| q_4$ $\texttt{s.t.}$ $q_1 \| q_3 \ \xi \ q_2' \| q_4 \wedge q_2' \| q_4 \overset{a}{\Longrightarrow} q_2'' \| q_4 \wedge q_1' \| q_3 \ \xi \ q_2'' \| q_4)$, and $(q_1 \| q_3, q_2' \| q_4)$ and $(q_1' \| q_3, q_2'' \| q_4)$ are in $\mathcal{S}$.
   (b) We have the same proof for the case $q_3 \xrightarrow{a}_3 q_3'$ and $\tilde{q} = q_1 \| q_3'$.

This compositionality result permits the modular design of the system into separate components. So, the attribute "modular" of the refinement relation η expresses the modular design ability as well as the partition of the refined system state space into modules.

7 Example

The definition of the modular refinement relation is motivated by the conditions expressed in terms of path set refinement as we saw in Section 4. We illustrate the refinement design with a simple example of a robot carrying parts inspired from [3]. It shows how the modules appear in the refined reachability graph, and how the non-determinism decreases during the refinement process. This last point justifies the clause non-determinism of Definition 6. The robot's example is voluntarily simple. We have studied the modular refinement relation about less trivial applications such as the protocol $T = 1$ [13], a BRP protocol [5] and so on.

We suppose the carrier device (CD) carries one part at a time from an arrival device AD located to its left towards two exit devices located respectively to the left (EDL) and the right (EDR) of the carrier device (see Figure 7).

The two refinement levels of transition systems (presented in Figure 8 and Figure 6 by their reachability graphs) model the transportation of parts by the carriage device on the exit devices. The first level ignores the movement of the carriage device and how parts arrive in it. The second level introduces the rotation movement of the carrier device.

At the first level of abstraction (see Figure 8), we are not concerned which of the exit devices is receiving the part. This introduces what we call an internal non-determinism in the abstract transition system. The variables are CD, EDL, EDR. Here, the invariant is well-typing: $CD, EDL, EDR \in \{free, busy\}$. Initially, all the devices are $free$. The actions that can be observed are the loading of a part (label L), the deposit of a part (label U), the exit of a part from the left exit device (label ExL) and the exit of a part from the right exit device (label ExR). The interpretation of the eight different states appears graphically. For example, $l(q_5)$ is $CD = busy \wedge EDL = free \wedge EDR = busy$. One can notice two transitions labeled by U originating from q_1 (internal non-determinism).

The second level of refinement (see Figure 6) introduces the observation of the two rotations, from the left to the right (RoR) and from the right to the left (RoL) that the carrier device must do in order to deposit a part either to the left or to the right. The glue invariant is $I_2 \overset{\text{def}}{=} (CD' = CD \wedge EDL' = EDL \wedge EDR' = EDR \wedge Pos \in \{l, r\})$.

A plausible behavior may remove the internal non-determinism by forcing the carrier device to unload on the left exit device if it is turned toward the left; respectively, to unload on the right exit device if it is turned toward the right (in position). Moreover, the rotation towards the left becomes a priority in the other situations since the arrival device is located to the left. This imposed behavior is supposed to minimize the number of rotations towards the right.

We verify that S_2 is a modular refinement of S_1 by substituting τ transitions by transitions labeled by RoL, RoR and verifying Definitions 5 and 6. Moreover, one can notice how the refinement introduces a partition of the refined transition system into modules. Notice in the example that the two transitions labeled by U exiting from the module of name q_1 are the reflection of the two transitions labeled by U exiting from q_1 at the abstract level. So, the traces containing U and beginning with q_1 remain. So, we have $readies(q_1) = readies(q'_{11}) \cup readies(q'_{12})$ where q'_{11} and q'_{12} are the two states in the module of name q_1.

Another plausible behavior may remove the internal non-determinism by simply forcing the carrier device to unload on the left (left first) when both devices are free. This imposed behavior is, as above, supposed to optimize the number of rotations since the arrival device is located to the left.

Notice that there is only one transition labeled by U exiting from the module q_1 which is the reflection of the two transitions labeled by U exiting from q_1 at the abstract level. So, some traces containing U and beginning by q_1 disappear between the abstract and the refined level but at least one trace containing U remains. We can use other strategies such as, for example, forcing the carrier device to unload on the exit device which has been freed the first (first free, first busy).

8 Application

As safety and liveness properties are essential for reactive systems, the verification uses different methods which are based either on the model or on the system description. The model-based verification methods use the labeled transition systems techniques equipped with logics or behavior equivalences. The methods which are working on the system description via its denotational semantics, prove theorems about the system behavior by using an appropriate logic. The theorem provers or the proof assistants are semi-automatic in contrast with the model-checkers, but they can handle infinite systems.

To get the advantages of both methods in verifying that the B event system satisfies its dynamic properties (expressed by LTL formulae), it is very interesting to be able to combine model-checking with theorem proving in the same verification tool set [8]. In [12, 14, 18], we propose an original combination of the above techniques. Proof is used when fully automatic (for example, tautology checker in propositional calculus). At the abstract level of the refinement, model-checking is used for dynamic property verification. The proposed refinement relation allows us to cross both techniques at the best of their advantages and possibilities. This is made possible because the proposed refinement relation determines modules in the refined reachability graph (for that, see [8]).

For finite state transition systems, the refinement verification and the module construction are implementable by an on-the-fly traversal of the refined transition system reachability graph. This is implemented as a component of a verification tool set [8].

9 Conclusion and Related Works

We are following an approach using both automatic-proof and model-checking technologies. This cooperation is permitted because of the refinement methodology [12, 14] which has been proposed to specify reactive systems and to verify their dynamic properties expressed in the LTL.

We have defined a formal framework for this methodology: the modular refinement relation between an abstract and a refined transition systems derived from the B design.

Since we apply the modular relation η to finitely branching transition systems, and since η does not accept the τ-divergence, it is closely related to the *refusal simulation* as defined by Ulidowsky in [24]. The refusal simulation is also a generalization of the ready simulation of Bloom et al. [10]. The main difference between the modular refinement relation η and the refusal simulation is in the non-determinism clause of Definition 6. This clause requires that q_1 relates to q_2, i.e., that the module of name q_1 exists but the refusal simulation does not. This constraint is the key to obtain the equality of the readies (see Theorem 2). Therefore, the modular refinement relation, which is also a generalization of the ready simulation, seems to be strictly included in the refusal simulation.

We can also relate the clauses of Definition 5 to the clauses $(a), (\epsilon), (O), (\Delta)$, (S) of van Glabbeek in [11] by taking into account of the fact that there is no silent moves in the abstract transition system. Therefore, we could situate the modular refinement relation as a *divergence sensitive stability respecting completed simulation* in the van Glabbeek' spectrum.

Another contribution of this work is that it gives a good semantics which authorizes us to compare the modular refinement relation with other refinement definitions [20, 15]. We anticipate that without the non-determinism, it is also equivalent to Lamport's TLA refinement.

One of the consequences of Theorem 4 is that the modular refinement relation is preserved w.r.t. the parallel composition. This allows us to keep the same refinement notion for both modular design and modular verification of reactive systems.

The partition of the refined system state space into modules is the second reason to call modular our refinement relation. It facilitates the verification by taking into account the system modularity introduced by the refinement. Accordingly we are currently building the verification tool set [8].

References

[1] J. R. Abrial. *The B Book*. Cambridge University Press - ISBN 0521-496195, 1996.

[2] J. R. Abrial. Extending B without changing it (for developing distributed systems). In *1st Conference on the B method*, pages 169–190, Nantes, France, November 1996.

[3] J. R. Abrial. Constructions d'automatismes industriels avec B. In *Congrès AFADL*, ONERA-CERT - Toulouse, France, May 1997. Invited lecture.

[4] J. R. Abrial, E. Bőrger, and H. Langmoeck. *Specifying and Programming the Steam Boiler Control*. LNCS 1165. Springer Verlag, 1996.

[5] J. R. Abrial and L. Mussat. Specification and design of a transmission protocol by successive refinements using B. LNCS, 1997.

[6] J. R. Abrial and L. Mussat. Introducing dynamic constraints in B. In *Second Conference on the B method*, LNCS 1393, pages 83–128, Montpellier, France, April 1998. Springer Verlag.

[7] P. Behm, P. Desforges, and J.M. Meynadier. MÉTÉOR: An industrial success in formal development. In *Second conference on the B method*, LNCS 1393, Montpellier, France, April 1998. Springer Verlag. Invited lecture.

[8] F. Bellegarde, J. Julliand, and H. Mountassir. Model-based verification through refinement of finite B event systems. In *Formal Method'99 B User Group Meeting*, CD-ROM publication, 1999.

[9] B. Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages*. PhD thesis, MIT, August 1989.

[10] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, January 1995.

[11] R. J. van Glabbeek. The linear time - branching time spectrum II: The semantics of sequential systems with silent moves. In *Proc. CONCUR'93, Hildesheim, Germany, LNCS 715*, pages 66–81. Springer-Verlag, August 1993.

[12] J. Julliand, F. Bellegarde, and B. Parreaux. De l'expression des besoins à l'expression formelle des propriétés dynamiques. *Technique et Science Informatiques*, 18(7), 1999.

[13] J. Julliand, B. Legeard, T. Machicoane, B. Parreaux, and B. Tatibouet. Specification of an integrated circuits card protocol application using B and linear temporal logic. In *Second conference on the B method*, LNCS 1393, pages 273–292, Montpellier, France, April 1998. Springer Verlag.

[14] J. Julliand, P.A. Masson, and H. Mountassir. Modular verification of dynamic properties for reactive systems. In *International Workshop on Integrated Formal Methods (IFM'99)*, York, Great Britain, 1999.

[15] L. Lamport. A temporal logic of actions. 16:872–923, May 1994.

[16] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6:1–35, January 1995.

[17] R. Milner. *Communication and Concurrency*. Prentice Hall Int., 1989.

[18] H. Mountassir, F. Bellegarde, J. Julliand, and P.A. Masson. Coopération entre preuve et model-checking pour vérifier des propriétés LTL. In *submission AFADL'2000*, 2000.

[19] O. Ore. Galois connections. *Trans. Amer. Math. Soc.*, (55):493–513, February 1944.
[20] A. Pnueli. System specification and refinement in temporal logic. In *Proc. 12th Conf. Found. of Software Technology and Theor. Comp. Sci., New Delhi, India, LNCS 652*, pages 1–38. Springer-Verlag, December 1992.
[21] L. E. Sanchis. Data types as lattices : retractions, closures and projections. *RAIRO Informatique Théorique et Applications*, 11(4):329–344, 1977.
[22] J. Sifakis. Property preserving homomorphisms of transition systems. In *Proc. Logics of Programs Workshop, Pittsburgh, LNCS 164*, pages 458–473. Springer-Verlag, June 1983.
[23] H. Treharne, J. Draper, and S. Schneider. Test case preparation using a prototype. In *Second conference on the B method*, LNCS 1393, pages 293–312, Montpellier, France, April 1998. Springer Verlag.
[24] I. Ulidowski. Equivalences on observable processes. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Sciences IEEE, New-York, IEEE Computer Society Press*, pages 148–161, 1992.

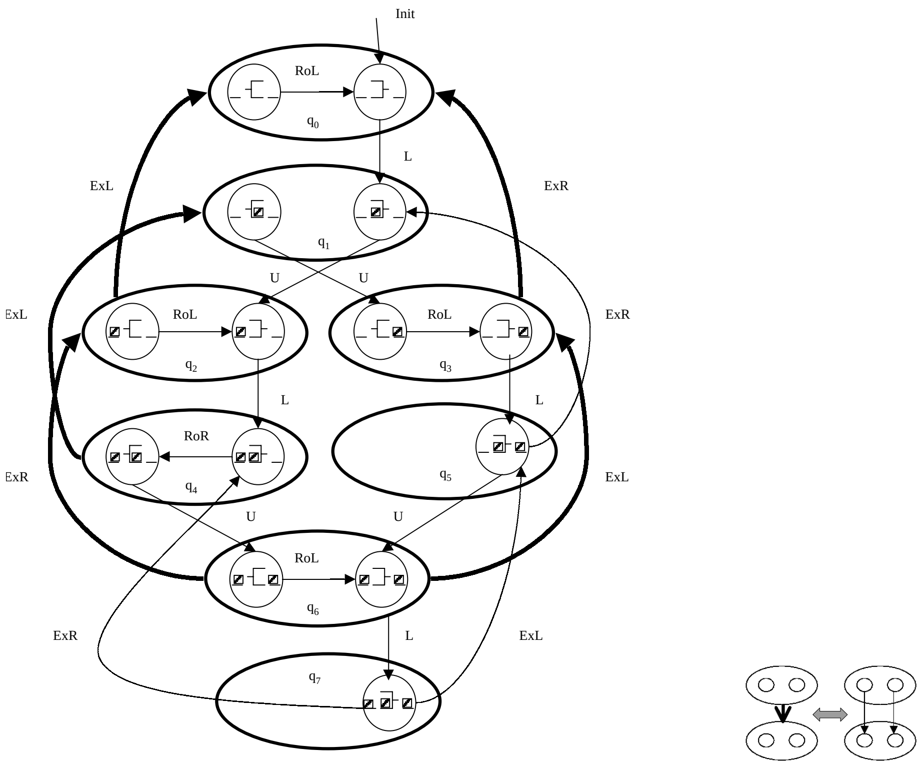

Fig. 6. A refined transition system (in position).

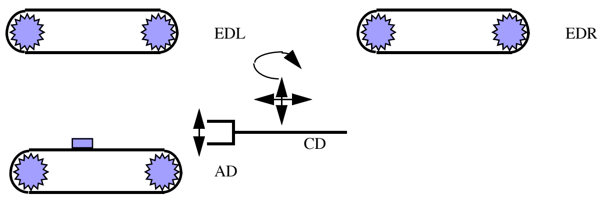

Fig. 7. The physical system.

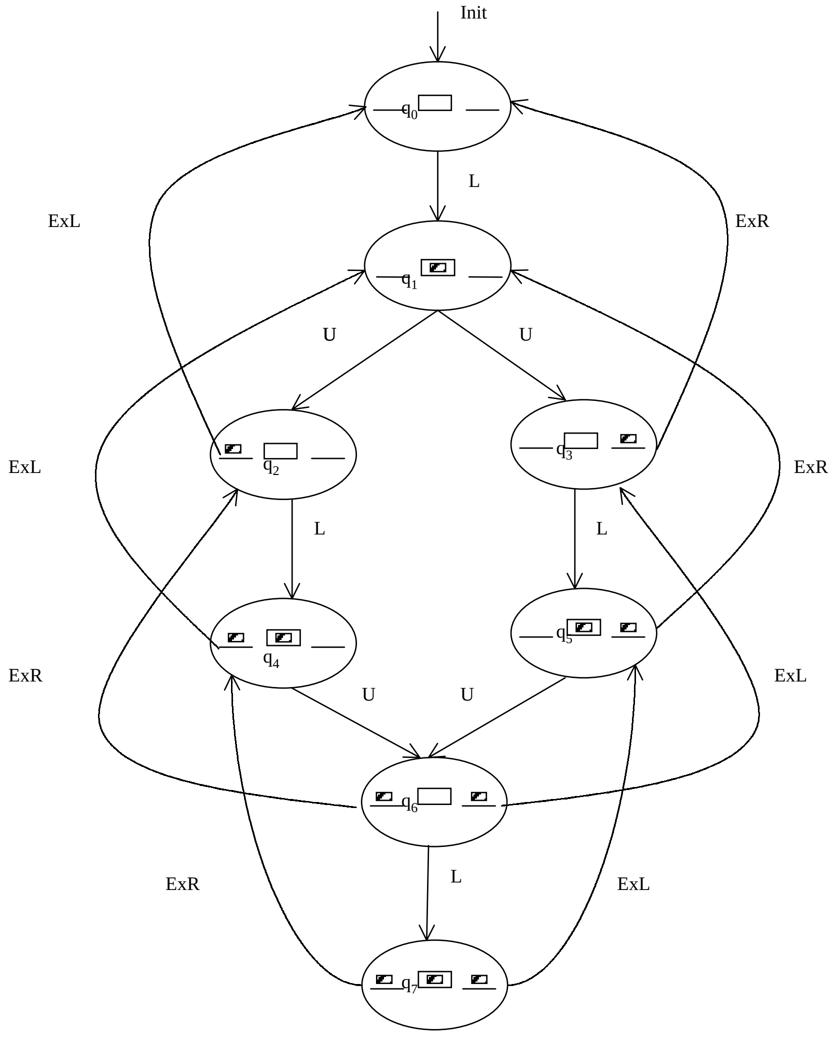

Fig. 8. The abstract transition system.