A General Model for Object Based Systems

Holger Naundorf
snoo@uni-paderborn.de

Abstract— **There is no commonly accepted general model for object based systems. Therefore different object based systems cannot be combined because their models are not compatible. We introduce a general model for timed and untimed object based systems. The use of this model for all object based systems enables the combination of independently developed systems.**

I. Introduction

To understand object based languages and systems it is important to have a model for these systems. Though there are some models, none of them is commonly accepted and general enough for modeling all object based systems.

Well known models are the actor model [2] and the layered semantics of POOL [3]. These models have the advantage that they model object based systems in an intuitive way, i.e. a system consists of objects that communicate with the help of a transmission unit. However, they have the disadvantage that timed object based systems cannot be modeled and that they are not hierarchical, i.e. object based systems do not represent objects and hence cannot be used as an object in an object based system. To avoid this problem, in [1] the concept of open systems is used, but this occurs to me as a work-around complicating the theory.

Other well known models are process calculi like CSP [6], CCS [11] or the π-calculus [13]. There are timed variants like RtCCS or DtCCS [16]. These models are hierarchical, but the objects communicate via handshaking which contradicts the encapsulation of objects – for the author the most important property of objects –, because two objects must know the internal state of each other before being able to communicate. The same is true for models for asynchronous communication based on process calculi like [7].

This paper introduces a general hierarchical model for object based systems, called *gMobS*. To call a model general one first has to define what the model is used for. The only basic property of object based systems is, that they consist of objects that communicate by messages only. Obviously, apart from systems usually called object based, circuits and neural nets meet this condition, too. This implies that the model must not only be a timed model but must be able to model continuous sending of messages. At first glance this seems awkward but in fact it does not really complicate the theory and unifies two different theories. Moreover, in the future some objects of object based systems might be implemented in hardware and continuous sending of messages could become important. Obviously systems without time can be modeled with timed models.

We concentrate mainly on the important results of this theory – many proofs are omitted or sketched only. A more detailed study can be found in [12].

II. Definitions

1. In this paper $\mathcal{T} \in \{\mathbb{N}_0, \mathbb{R}_+\}$ models time.
2. For $t_1, t_2 \in \mathcal{T}$ let $\mathcal{T}_{t_1}^{t_2} = \{t \in \mathcal{T} \mid t_1 \leq t < t_2\}$ be the set of all times between t_1 and t_2.
3. For an arbitrary set A and $f \in (\mathcal{T} \rightarrow A)$ let $f_{t_1}^{t_2} \in (\mathcal{T}_{t_1}^{t_2} \rightarrow A)$ with:
$\forall t \in \mathcal{T}_{t_1}^{t_2} : f(t) = f_{t_1}^{t_2}(t)$
4. For an arbitrary set A let $f, g \in (\mathcal{T} \rightarrow A)$ and $t_0 \in \mathcal{T}$, then f equals g up to time t_0, $f =_{t_0} g :\Leftrightarrow \forall t \in \mathcal{T}, t < t_0 : f(t) = g(t)$
5. Consider a relation $R \subset A \times B$, then $Do(R) = \{a \in A \mid \exists b \in B : (a, b) \in R\}$ is the domain of R.
6. For any set A, $\mathcal{P}^E(A) = \{F \subset A \mid |F| < \infty\}$ is the set of all finite subsets of A.
7. For any set A, $f \in (\mathcal{T} \rightarrow A)$ and $\tau \in A$, $\mathcal{MT}_\tau(f) = \{t \in \mathcal{T} \mid f(t) \neq \tau\}$.

III. Objects

A universal model for object based systems requires a universal model for objects. Moreover, a model for objects should respect their most basic property – they hide their implementation and their state (they even hide, if they have a state).

The actor-model [2] and the layered semantics of POOL [3] can only describe a very limited set of objects. Systems of objects are not objects themselves. Process calculi like CSP [6], CCS [11] or the π-calculus [13] do not respect encapsulation of objects.

A suitable mechanism are (abstract) time systems introduced in system- and control theory [10]. Time systems are a generalization of finite traces [6], infinite traces [8], and timed traces [17].

A. I/O-descriptions

Definition:

1. A *time system* (see [10] Def. 5.2) is a triple $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ where $\mathcal{I} \neq \emptyset$ is an arbitrary set representing the set of messages possibly received, $\mathcal{O} \neq \emptyset$ is an arbitrary set representing the set of messages possibly sent and $\mathcal{U} \subset (\mathcal{T} \to \mathcal{I}) \times (\mathcal{T} \to \mathcal{O})$ is a set representing the behavior of the system.
2. An object provides a behavior for every input. An *I/O-description* is a time system $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ where $\mathcal{U} = \emptyset$ or $\mathcal{U}$ is a *total* relation:
   $\forall I \in (\mathcal{T} \to \mathcal{I})\, \exists O \in (\mathcal{T} \to \mathcal{O}) : (I, O) \in \mathcal{U}$
   If $\mathcal{U}$ is total, then $\mathcal{D}$ is called *real*.
3. $\mathcal{DS}_{\mathcal{I}}^{\mathcal{O}}$ is the set of all I/O-descriptions $(\mathcal{I}, \mathcal{O}, \mathcal{U})$.
   The I/O-description $\top := \top_{\mathcal{I}}^{\mathcal{O}} := (\mathcal{I}, \mathcal{O}, \emptyset)$ is needed for $\mathcal{DS}_{\mathcal{I}}^{\mathcal{O}}$ forming a lattice.

□

This formalism implies that an I/O-description receives and sends messages all the time, while objects used in object based systems send and receive messages sometimes only. Non-receiving (non-sending) of messages can be interpreted as receiving (sending) messages with no information. Most formalisms simply neglect these empty messages since they are considered as useless.

An important example of a timeless object is a buffer:

Example: Let N be an arbitrary set and $\tau \notin N$. $N_\tau = N \cup \{\tau\}$. Let $\phi \subset N \times N$.

Then $\mathcal{D}_{Buf(N_\tau,\phi)} = (N_\tau, N_\tau, \mathcal{U}_{Buf(N_\tau,\phi)})$ with $\mathcal{U}_{Buf(N_\tau,\phi)} = \{(I, O) \in (\mathcal{T} \to N_\tau) \times (\mathcal{T} \to N_\tau) \mid \exists f : \mathcal{MT}_\tau(I) \to \mathcal{MT}_\tau(O)$ with

1. f is bijective [no messages are lost or duplicated]
2. $\forall t \in \mathcal{MT}_\tau(I) : t \leq f(t)$ [a message is only sent after being received]
3. $\forall t \in \mathcal{MT}_\tau(I) : I(t) = O(f(t))$ [no message is changed]
4. $\forall t_1, t_2 \in \mathcal{MT}_\tau(I)$:
   $(f(t_1) < f(t_2) \Rightarrow t_1 < t_2 \vee (I(t_1), I(t_2)) \in \phi$
   [a message is only sent earlier than another message, if it was received earlier or it was allowed to pass (by ϕ)]

$\mathcal{D}_{Buf(N_\tau,\phi)}$ is a buffer, that stores messages not equal to τ and sends them (without any changes) at a time which is not earlier than the receiving of them. A message n_1 is sent earlier than a message n_2 only if n_1 arrived earlier than n_2 or if n_1 is allowed to pass n_2, which is determined by $(n_1, n_2) \in \phi$.

This means that $\mathcal{D}_{Buf(N_\tau,\emptyset)}$ is a buffer which sends messages in exactly the same order as they arrive. In contrast $\mathcal{D}_{Buf(N_\tau,N\times N)}$ is a buffer, which sends all received messages in an arbitrary order.

A buffer can receive infinite many messages not equal to τ in a finite time interval. □

B. Determinism

An I/O-description that can answer with exactly one answer to every input is a special case. In reality this is almost never the case, because objects do not possess an exact clock. The condition for determinism is very restrictive, because time is considered. Objects, that are determined intuitively, usually are not determined according to the following definition. It is possible to define this kind of determinism in this model, too. But for that some messages representing empty messages must be marked and then determinism can be defined by the possible orders of the remaining messages.

Definition: A time system $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ is called *determined* or *functional*, if $\mathcal{U}$ is functional: $\forall I \in (\mathcal{T} \to \mathcal{I})\, \forall (I, O_1), (I, O_2) \in (\mathcal{T} \to \mathcal{O})$: $O_1 = O_2$.

In all other cases $\mathcal{D}$ is called not determined. □

In contrast to [10] throughout this paper the notion determined is used, because it seems more intuitive than functional.

Usually an observer is not interested, if an object is really able to answer with all the messages, (s)he expects. Moreover the observer does not have the chance to decide, if an object is able to answer with all messages (s)he expects.

Definition: Let $\mathcal{D}_1 = (\mathcal{I}_1, \mathcal{O}_1, \mathcal{U}_1)$ and $\mathcal{D}_2 = (\mathcal{I}_1, \mathcal{O}_1, \mathcal{U}_2)$ be I/O-descriptions.

$\mathcal{D}_1$ is called more determined than $\mathcal{D}_2$, $\mathcal{D}_1 \prec \mathcal{D}_2$, if: $\mathcal{U}_1 \subset \mathcal{U}_2$. □

Though the relation 'more determined' is a canonical relation, it seems not to be introduced in system theory. Probably this is due to the fact, that this relation is only meaningful, if the definition sets of the compared time systems are equal.

Remark: Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be real I/O-descriptions with $\mathcal{D}_1 \prec \mathcal{D}_2$. If $\mathcal{D}_2$ is determined, then $\mathcal{D}_1 = \mathcal{D}_2$. □

C. Lattice-Structure

Remark: The relation $\prec$ is transitive, reflective and antisymmetric and hence a partial order. □

Definition: Let S be a set and let $\mathcal{D}_s = (\mathcal{I}_s, \mathcal{O}_s, \mathcal{U}_s) \in \mathcal{DS}_{\mathcal{I}}^{\mathcal{O}}$ for $s \in S$.

1. Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ with: $\mathcal{U} = \bigcup_{s \in S} \mathcal{U}_s$. Then $\mathcal{D} = \bigcup_{s \in S} \mathcal{D}_s$ is the union of the $\mathcal{D}_s$.
2. Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ with:
$$\mathcal{U} = \begin{cases} \bigcap_{s \in S} \mathcal{U}_s & : \quad \bigcap_{s \in S} \mathcal{U}_s \text{ is total} \\ \emptyset & : \quad \text{otherwise} \end{cases}$$
Then $\mathcal{D} = \bigcap_{s \in S} \mathcal{D}_s$ is the intersection of the $\mathcal{D}_s$.

□

Lemma: Let S be a set and $D = \{\mathcal{D}_s \mid s \in S\} \subset \mathcal{DS}_{\mathcal{I}}^{\mathcal{O}}$.

1. $\bigcap_{s \in S} \mathcal{D}_s$ is the least upper bound of D.
2. $\bigcup_{s \in S} \mathcal{D}_s$ is the greatest lower bound of D.

□

Remark: $(\mathcal{DS}_{\mathcal{I}}^{\mathcal{O}}, \prec)$ is a complete lattice with:

1. $\bigsqcup = \bigcap$
2. $\sqcap = \bigcup$
3. $\bot = (\mathcal{I}, \mathcal{O}, (\mathcal{T} \to \mathcal{I}) \times (\mathcal{T} \to \mathcal{O}))$
4. $\top = (\mathcal{I}, \mathcal{O}, \emptyset)$

□

D. Properties

The definition of I/O-descriptions is very general. There exist many I/O-descriptions, that do not describe a meaningful object. This section defines classes of I/O-descriptions, which are meaningful – especially in context with object based systems.

An object based system consists of a set of objects – defined by I/O-descriptions –, which can communicate with the help of a transmission unit. The transmission unit itself is defined by an I/O-description. The input into the transmission unit consists of the output of the objects and an input coming from the outside. The output of the transmission unit consists of the input into the objects and an output directed to the outside.

To develop a general model, the input from and the output to the outside should be considered independently from the input from and the output to the objects.

Definition: Let X be a set and for $x \in X$ let $W(x)$ be a set. Let $Z \subset X$, then

1. $Int_Z(\prod_{x \in X} W(x)) = \prod_{x \in X \setminus Z} W(x)$
2. $Int_Z : (\prod_{x \in X} W(x)) \to \prod_{x \in X \setminus Z} W(x)$, $(s_x)_{x \in X} \mapsto (s_x)_{x \in X \setminus Z}$
3. $Ext_Z(\prod_{x \in X} W(x)) = \prod_{x \in Z} W(x)$
4. $Ext_Z : (\prod_{x \in X} W(x)) \to \prod_{x \in Z} W(x)$, $(s_x)_{x \in X} \mapsto (s_x)_{x \in Z}$

□

Pre-Causal

Let $\mathcal{D}_{NP} = (\mathcal{I}_{NP}, \mathcal{O}_{NP}, \mathcal{U}_{NP})$ with $\mathcal{T} = \mathbb{N}_0$, $\mathcal{I}_{NP} = \mathcal{O}_{NP} = \{0, 1\}$ and $\mathcal{U}_{NP} = \{(I, O) \mid \forall n \in \mathbb{N}_0 : O(n) = I(n+1)\}$. The output of $\mathcal{D}_{NP}$ at time n is the input at time $n+1$. Every time $\mathcal{D}_{NP}$ must know what message it will receive next. In reality such an object cannot exist.

An I/O-description is called pre-causal, if at any time the output only depends on the received input up to that time. Since the behavior is not defined by calculations, it is not possible to find a causal connection between input and output. But one can say that the output of an I/O-description up to a time t_0 only depends on the input up to t_0, if for all inputs, that are equal up to t_0, the same output up to t_0 is possible (see [10] Def. 2.4).

Causal

Though the pre-causal I/O-descriptions are a quite nice class, they are too general to be handled. Therefore this notion is strengthened. An I/O-description is called causal, if it is the union of determined pre-causal I/O-descriptions. In [10] Def. 2.1 causal is defined by completely different means. However, both definitions are equivalent.

Pre-Delayed

For investigating object based systems, it is important to identify a huge class of object based systems, that themselves represent real pre-causal I/O-descriptions. If the objects and the transmission unit are only guaranteed to be causal, the I/O-description represented by this object based system can neither be guaranteed to be real nor pre-causal.

But if the transmission unit can continue its output for a short time without knowing the input coming from the objects during this time, the transmission unit can always continue a calculation a bit and by this produce a calculation.

The transmission unit must only be able to continue its output to the objects in this way – the transmission unit needs to be internally pre-delayed only.

Delayed

Similar to pre-causal I/O-descriptions the class of pre-delayed I/O-descriptions is not restricted enough to be handled. Therefore this notion is strengthened, too. An I/O-description is called delayed if it is the union of determined pre-delayed I/O-descriptions. Obviously the definition is analogous to the definition of causal. An analogous definition for delayed based on the definition of causal given in [10] is not known to the author.

Definition: Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ be a time system. $\mathcal{D}$ is called

1. *pre-causal*, if:
   $\forall (I,O) \in \mathcal{U}\, \forall t \in \mathcal{T}\, \forall \hat{I} \in Do(\mathcal{U})$ with $I =_t \hat{I}$
   $\exists (\hat{I}, \hat{O}) \in \mathcal{U} : O =_t \hat{O}$
2. *causal*, if there are a set S and determined pre-causal time systems $\mathcal{D}_s$ for $s \in S$ with:
   $\mathcal{D} = \bigcup_{s \in S} \mathcal{D}_s$
3. internally *pre-delayed* with respect to (Z_I, Z_O), if $\mathcal{D}$ is pre-causal with:
   $\forall (I,O) \in \mathcal{U}\, \forall t \in \mathcal{T}\, \exists t < t_0 \in \mathcal{T}\, \forall \hat{I} \in Do(\mathcal{U})$
   with $I =_t \hat{I}$ and $Ext_{Z_I}(I) = Ext_{Z_I}(\hat{I})$
   $\exists (\hat{I}, \hat{O}) \in \mathcal{U}$:
   $O =_t \hat{O}$ and $Int_{Z_O}(O) =_{t_0} Int_{Z_O}(\hat{O})$
   $\mathcal{D}$ is called pre-delayed if it is internally pre-delayed with respect to $(\emptyset, \emptyset)$.
4. internally *delayed* with respect to (Z_I, Z_O), if there are a set S and determined pre-delayed time systems $\mathcal{D}_s$ for $s \in S$ with:
   $\mathcal{D} = \bigcup_{s \in S} \mathcal{D}_s$
   $\mathcal{D}$ is called delayed if it is delayed with respect to $(\emptyset, \emptyset)$.

□

E. Remarks on Properties

Remark: Let S be a set and $\mathcal{D}_s$ be time systems for $s \in S$. Let $\mathcal{D} = \bigcup_{s \in S} \mathcal{D}_s$.

1. $\forall s \in S : \mathcal{D}_s$ is pre-causal $\Rightarrow \mathcal{D}$ is pre-causal
2. $\forall s \in S : \mathcal{D}_s$ is causal $\Rightarrow \mathcal{D}$ is causal
3. $\forall s \in S : \mathcal{D}_s$ is internally pre-delayed with respect to (Z_I, Z_O)
   $\Rightarrow \mathcal{D}$ is internally pre-delayed with respect to (Z_I, Z_O)
4. $\forall s \in S : \mathcal{D}_s$ is internally delayed with respect to (Z_I, Z_O)
   $\Rightarrow \mathcal{D}$ is internally delayed with respect to (Z_I, Z_O)

□

Proof: Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ and for $s \in S$ let $\mathcal{D}_s = (\mathcal{I}_s, \mathcal{O}_s, \mathcal{U}_s)$.

1. Consider $(I,O) \in \mathcal{U}$, $t \in \mathcal{T}$, and $\hat{I} \in (\mathcal{T} \to \mathcal{I})$ with $I =_t \hat{I}$. Then there exists a $s \in S$ with $(I,O) \in \mathcal{U}_s$. Since $\mathcal{D}_s$ is pre-causal, there exists $(\hat{I}, \hat{O}) \in \mathcal{U}_s \subset \mathcal{U}$ with $O =_t \hat{O}$. Hence $\mathcal{D}$ is pre-causal.
2. The other cases are obvious or can be proved analogously.

□

Remark: Let $\mathcal{D}$ be a time system.

1. $\mathcal{D}$ causal $\Rightarrow \mathcal{D}$ pre-causal
2. $\mathcal{D}$ pre-delayed $\Rightarrow \mathcal{D}$ pre-causal
3. $\mathcal{D}$ delayed $\Rightarrow \mathcal{D}$ pre-delayed and causal

□

Proof: Union preserves the properties and a pre-delayed I/O-description is pre-causal by definition. □

Example: The buffer $\mathcal{D}_{Buf(N_\tau, \phi)}$ is delayed and hence pre-delayed, causal and pre-causal.

While it is easy to show, that $\mathcal{D}_{Buf(N_\tau,\phi)}$ is pre-delayed and hence pre-causal, it is a lot more work to show, that $\mathcal{D}_{Buf(N_\tau,\phi)}$ is delayed or causal.

[10] Prop. 2.1 provides conditions for an I/O-description to be causal. In [10] it is said that these conditions are fulfilled in most cases. But even the slight generalization [12] Satz 4.35 of [10] Prop. 2.1 is not strong enough for $\mathcal{D}_{Buf(N_\tau,\phi)}$ and for most untimed objects. This illustrates the difference between system theory and the theory for object based systems. $\square$

Remark: Let $\mathcal{T} = \mathbb{N}_0$, $\mathcal{D}_1 = (\{0,1\}, \{0,1\}, \{(I,O) \mid O(0) = 0 \wedge \forall n \in \mathbb{N}_0 : O(n+1) = I(n)\})$ and $\mathcal{D}_2 = \perp_{\{0,1\}}^{\{0,1\}}$.

$\mathcal{D}_1$ and $\mathcal{D}_2$ are pre-causal, causal, pre-delayed and delayed.

Let $\mathcal{D}_3 = \mathcal{D}_1 \cup (\{0,1\}, \{0,1\}, \{(I,O) \mid \forall n \in \mathbb{N}_0 : O(n) = I(n+1)\})$. Then $\mathcal{D}_1 \prec \mathcal{D}_3$ and $\mathcal{D}_3 \prec \mathcal{D}_2$, but $\mathcal{D}_3$ is neither pre-causal, causal, pre-delayed nor delayed.

It seems that the $\prec$ relation does not possess any interrelations with the defined properties. $\square$

F. Other Characterizations of Causal

The definition of causal seems to be arbitrary and it is interesting to know if there exist other characterizations.

[10] Def. 2.1 defines causal in a completely different way. It is not obvious that both definitions have something in common. In [10] Prop. 2.8 it is shown that a time system is causal if and only if it has a state space representation ([10] Def. 2.8). The same is true for causal I/O-descriptions and consequently both definitions are equivalent.

Lemma: Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ be an I/O-description. $\mathcal{D}$ is causal if and only if there exist

1. a set $States_t$ of states for every $t \in \mathcal{T}$
2. a set $SStates \subset States_0$ of starting states
3. a state transition function $TransFunc_{t_1}^{t_2} \in (States_{t_1} \times (\mathcal{T}_{t_1}^{t_2} \to \mathcal{I}) \to States_{t_2})$ for all $t_1, t_2 \in \mathcal{T}$ with $t_1 \leq t_2$.
4. an output function $OutFunc_t \in ((States_t \times \mathcal{I}) \to \mathcal{O})$ for all $t \in \mathcal{T}$

such that

1. $\forall t_1, t_2, t_3 \in \mathcal{T}, t_1 \leq t_2 \leq t_3$ $\forall c \in States_{t_1} \forall I \in (\mathcal{T} \to \mathcal{I})$: $TransFunc_{t_2}^{t_3}(TransFunc_{t_1}^{t_2}(c, I_{t_1}^{t_2}), I_{t_2}^{t_3}) = TransFunc_{t_1}^{t_3}(c, I_{t_1}^{t_3})$
2. $\forall t \in \mathcal{T} \forall c \in States_t \forall I \in (\mathcal{T}_t^t \to \mathcal{I})$: $TransFunc_t^t(c, I) = c$
3. $\forall (I,O) \in (\mathcal{T} \to \mathcal{I}) \times (\mathcal{T} \to \mathcal{O})$: $(I,O) \in \mathcal{U} \Leftrightarrow \exists c \in SStates \,\forall t \in \mathcal{T}$: $OutFunc_t(TransFunc_0^t(c, I_0^t), I(t)) = O(t)$

$\square$

A similar characterization of causal is the notion of oraclizable I/O-descriptions (similar to [14]).

Remark: Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ be an I/O-description. $\mathcal{D}$ is causal if and only if there exist sets OS and $OI \subset (\mathcal{T} \to OS)$ and a determined, pre-causal I/O-description $\mathcal{D}_{Or} = (\mathcal{I} \times OS, \mathcal{O}, \mathcal{U}_{Or})$ such that:
$\mathcal{U} = \{(I,O) \mid \exists I_0 \in OI : ((I, I_0), O) \in \mathcal{U}_{Or}\}$. $\square$

IV. Partial Objects

It is very laborious to define the complete behavior of an I/O-description. Moreover in literature objects cannot answer to every input, i.e. they are partially defined only.

Definition: Let $\mathcal{D}^P = (\mathcal{I}, \mathcal{O}, \mathcal{U}^P)$ be a time system. An I/O-description $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ is called the canonical I/O-description for the time system $\mathcal{D}^P$, if $\mathcal{U}^P \subset \mathcal{U}$ and $\mathcal{D} = \bigcup_{\mathcal{D}_d \in D} \mathcal{D}_d$ with
$D = \{\mathcal{D}_1 = (\mathcal{I}_1, \mathcal{O}_1, \mathcal{U}_1) \in \mathcal{DS}_{\mathcal{I}}^{\mathcal{O}} \mid$

1. $\mathcal{D}_1$ is pre-causal
2. $\forall (I,O) \in \mathcal{U}_1 \forall t \in \mathcal{T} \cup \{\infty\}$: $(\exists \hat{I} \in Do(\mathcal{U}^P)$ with $I =_t \hat{I}$ $\Rightarrow \exists (I_1, O_1) \in \mathcal{U}^P : (I_1, O_1) =_t (I, O))$

$\}$.

If there does not exist such an I/O-description, then $\top_{\mathcal{I}}^{\mathcal{O}}$ is called the canonical I/O-description for $\mathcal{D}^P$. $\square$

Remark: Let $\mathcal{D}^P$ be a time system with canonical I/O-description $\mathcal{D}$.

1. If $\mathcal{D}^P$ is pre-causal, then $\mathcal{D}$ is real and pre-causal.
2. If $\mathcal{D}^P$ is causal, then $\mathcal{D}$ is causal.
3. If $\mathcal{D}^P$ is not pre-causal, then $\mathcal{D} = \top$.
4. If $\mathcal{D}^P$ is a pre-causal I/O-description, then $\mathcal{D} = \mathcal{D}^P$.

$\square$

Remark: Let $\mathcal{D}_1^P = (\mathcal{I}_1, \mathcal{O}_1, \mathcal{U}_1^P)$ and $\mathcal{D}_2^P = (\mathcal{I}_1, \mathcal{O}_1, \mathcal{U}_2^P)$ be pre-causal time systems with canonical I/O-descriptions $\mathcal{D}_1$ and $\mathcal{D}_2$. If

$Do(\mathcal{U}_1^P) = Do(\mathcal{U}_2^P)$ and $\mathcal{U}_1^P \subset \mathcal{U}_2^P$, then $\mathcal{D}_1 \prec \mathcal{D}_2$. □

V. Object Based Systems

Having defined objects it is possible to proceed to object based systems. An object based system is similar to a system in the actor model [2] or the model for the layered semantics of POOL [3], because a system consists of a set of objects which communicate with the help of a transmission unit. Like in the layered semantics of POOL the communication with the outside world is modeled by communicating with a special object (the object $\perp$).

In contrast to the actor and POOL model, objects do not have the ability to create new objects, because it is difficult to regard creation of new objects as sending of messages. Instead an object based system may consist of infinite many objects – similar to a common technique for Petri nets [5]. The 'creation' of a new object can be modeled in two different ways:

1. The 'creating' object chooses an object, that is not used already, and initializes it. This requires that two different objects do not try to initialize the same object. Therefore a space of addresses similar to the actor model [2] is needed.
2. The 'creating' object sends a message to the transmission unit, which chooses an unused object and initializes it. This technique is more similar to the actually implemented strategies within computers.

Object based systems have some similarity with two level systems in system theory [9], though these systems are finite and the controllability in other settings is investigated.

It is important to notice that object based systems are only a means for describing I/O-descriptions.

A. Object Based Systems

Definition: A tuple $\mathcal{S} = (\mathcal{IM}, \mathcal{OM}, \Omega, \mathcal{M}^I, \mathcal{M}^O, \mathcal{OD}, \mathcal{OT})$ is called *object based system*, if:

1. $\mathcal{IM} \neq \emptyset$ is the set of messages the system can receive from the outside, $\mathcal{OM} \neq \emptyset$ is the set of messages the system can send to the outside.
2. Ω is the set of objects of the system. To communicate with the outside the system uses an object $\perp \notin \Omega$. Sometimes the transmission unit is handled like an object $T \notin \Omega$. $\hat{\Omega} := \Omega \cup \{\perp\}$, $\Omega^T := \Omega \cup \{T\}$ and $\hat{\Omega}^T := \Omega \cup \{T, \perp\}$.
3. If $\omega \in \Omega$ then $\mathcal{M}^I(\omega)$ is the set of messages that can be sent to the object ω and $\mathcal{M}^O(\omega)$ is the set of messages that can be sent from the object ω.
$\mathcal{M}^I(\perp) = \mathcal{IM}$ and $\mathcal{M}^O(\perp) = \mathcal{OM}$.
4. If $\omega \in \Omega$ then $\mathcal{OD}(\omega)$ specifies the object ω by an I/O-description.
$\mathcal{OD}(\perp) := \perp_{\mathcal{IM}}^{\mathcal{OM}}$
5. $\mathcal{OD}(T) := \mathcal{OT} = (\prod_{\omega \in \hat{\Omega}} \mathcal{IM}(\omega), \prod_{\omega \in \hat{\Omega}} \mathcal{OM}(\omega), \mathcal{U}_T)$

□

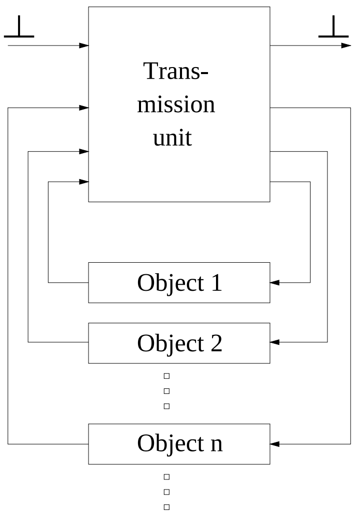

Fig. 1. Object Based System

Example: Let $\mathcal{T} = \mathbb{N}_0$ and $t_F \in \mathcal{T}$. Let $\mathcal{D}_F = (\mathcal{I}_F, \mathcal{O}_F, \mathcal{U}_F)$ with:

1. $\mathcal{IM}_F = \{S, R, N\}$ and $\mathcal{OM}_F = \{0, 1\}$.
2. $\Omega_F = \{X\}$
3. $\mathcal{M}_F^I(X) = \mathcal{M}_F^O(X) = \{0, 1\}$
4. $\mathcal{OD}_F(X) = (\{0,1\}, \{0,1\}, \{(I, O) \mid I = O\})$
5. $\mathcal{OT}_F = (\{S, R, N\} \times \{0,1\}, \{0,1\}^2, \{((I_\perp^T, I_X^T), (O_\perp^T, O_X^T)) \in (\mathcal{T} \to \{S, R, N\} \times \{0,1\}) \times (\mathcal{T} \to \{0,1\}^2) \mid O_\perp^T = I_X^T \wedge \forall t \in \mathcal{T}:$

$$O_X^T(t) = \begin{cases} 0 & : \text{ if } I_\perp^T(t) = R \\ 1 & : \text{ if } I_\perp^T(t) = S \\ I_X^T(t - t_F) & : \text{ if } I_\perp^T(t) = N \wedge t \geq t_F \\ 0 & : \text{ otherwise} \end{cases}$$

})

This object based system represents the easiest circuit, that implements a storage unit. Below it will be called Flip.

From the outside the system accepts the messages S for set, R for reset and N for nothing or no influence.

$\mathcal{OT}_F$ is internally pre-delayed with respect to $(\perp, \perp)$ if and only if $t_F > 0$. $\mathcal{OT}_F$ is not pre-delayed for any t_F.

The object and the transmission unit are determined and pre-causal, hence causal. □

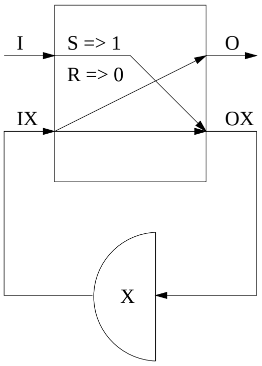

Fig. 2. Flip

B. *Object Based Systems as I/O-Descriptions*

When an object based system calculates, every object and the transmission unit calculate for themselves. The transmission unit gets the output of the objects and an input from the outside as input and must calculate according to its I/O-description. The transmission unit produces output which is used as input into the objects and as output to the outside. The objects behave according to their I/O-description and the given input.

Consequently a calculation consists of pairs of input/output for each object (including $\perp$ and T). Moreover the inputs and outputs must fit together.

Definition: Let $\mathcal{S} = (\mathcal{IM},\ \mathcal{OM},\ \Omega,\ \mathcal{M}^I,\ \mathcal{M}^O,\ \mathcal{OD},\ \mathcal{OT})$ be an object based system with $\mathcal{OD}(\omega) = (\mathcal{I}_\omega, \mathcal{O}_\omega, \mathcal{U}_\omega)$ for $\omega \in \hat{\Omega}^T$. Let $I \in (\mathcal{T} \to \mathcal{IM})$ and $O \in (\mathcal{T} \to \mathcal{OM})$.
$\mathcal{R} = (((I_\omega, O_\omega))_{\omega \in \hat{\Omega}}, (I_T, O_T)) \in (\prod_{\omega \in \hat{\Omega}} \mathcal{U}_\omega) \times \mathcal{U}_T$ [sometimes the entry $(I_\perp, O_\perp)$ is omitted] is called *calculation* from $\mathcal{S}$ for input I with output O up to $t_0 \in \mathcal{T} \cup \{\infty\}$, if $I_T =_{t_0} \prod_{\omega \in \hat{\Omega}} O_\omega$, $O_T =_{t_0} \prod_{\omega \in \hat{\Omega}} I_\omega$, $I_\perp =_{t_0} O$ and $O_\perp =_{t_0} I$.

If $\mathcal{R}$ is a calculation up to ∞ it is just called calculation. □

Example: Some examples of the start of calculations of a Flip with different delays t_F will be shown. Let
$\mathcal{R} = ((I_\perp, O_\perp), (I_X, O_X), ((I_\perp^T, I_X^T), (O_\perp^T, O_X^T))$ be a calculation. Then $I = O_\perp$ represents the input coming from the outside and $O = I_\perp$ represents the output sent to the outside. According to the definition of calculation and of the Flip, it is: $O = I_\perp = O_\perp^T = I_X^T = O_X = I_X = O_X^T$.

1. Let $t_F = 0$. The beginning of two different calculations for the same input looks like:

time	0	1	2	3	4	5	6	7
I	N	N	N	R	N	N	S	N
O	0	0	1	0	1	0	1	0

time	0	1	2	3	4	5	6	7
I	N	N	N	R	N	N	S	N
O	1	1	0	0	0	1	1	1

   Obviously there exist different calculations with different outputs for one input, though the object and the transmission unit are determined and pre-causal. The Flip does not store anything.

2. Let $t_F = 1$.

time	0	1	2	3	4	5	6	7
$I_\perp$	N	N	N	R	N	N	S	N
$O_\perp$	0	0	0	0	0	0	1	1

   In this case there exists only one calculation and the Flip stores its content.

□

Definition: Let $\mathcal{S} = (\mathcal{IM},\ \mathcal{OM},\ \Omega,\ \mathcal{M}^I,\ \mathcal{M}^O,\ \mathcal{OD},\ \mathcal{OT})$ be an object based system. Let $\mathcal{D}_\mathcal{S} = (\mathcal{I}_\mathcal{S}, \mathcal{O}_\mathcal{S}, \mathcal{U}_\mathcal{S})$ with:

1. $\mathcal{I}_\mathcal{S} = \mathcal{IM}$ and $\mathcal{O}_\mathcal{S} = \mathcal{OM}$
2. $\mathcal{U}_\mathcal{S} = \{(I, O) \mid$ it exists a calculation of $\mathcal{S}$ for input I with output $O\ \}$

If $\mathcal{D}_\mathcal{S}$ is an I/O-description, then $\mathcal{D}_\mathcal{S}$ is called the canonical I/O-description of $\mathcal{S}$. Otherwise $\top$ is called the canonical I/O-description of $\mathcal{S}$. □

Below some examples illustrate that the canonical I/O-description of an object based system with causal objects and transmission unit need not behave well. The canonical I/O-description needs neither to be real nor determined, even if the objects and the transmission unit are determined, nor pre-causal.

Example:

1. Let $t_F = 0$, then the canonical I/O-description of the Flip is:
   $\mathcal{D}_{\mathcal{S}_F} = (\{S, R, N\}, \{0,1\}, \{(I,O) \mid$
   (a) $I(t) = S \Rightarrow O(t) = 1$
   (b) $I(t) = R \Rightarrow O(t) = 0$
   $\})$.
   $\mathcal{D}_{\mathcal{S}_F}$ is not determined.
2. Let $t_F = 1$, then the canonical I/O-description of the Flip is:
   $\mathcal{D}_{\mathcal{S}_F} = (\{S, R, N\}, \{0,1\}, \{(I,O) \mid$
   $$O(t) = \begin{cases} 0 & : \text{ if } A_t = \emptyset \\ 1 & : \text{ if } I(\max(A_t)) = S \\ 0 & : \text{ if } I(\max(A_t)) = R \end{cases}$$
   with $A_t = \{\hat{t} \mid \hat{t} \leq t \wedge I(\hat{t}) \neq N\}$ $\})$.
   $\mathcal{D}_{\mathcal{S}_F}$ is determined.

□

Example: Let $\mathcal{D}_I = (\mathcal{I}_F, \mathcal{O}_F, \mathcal{U}_F)$ with:

1. $\mathcal{IM}_I = \mathcal{OM}_I = \{0,1\}$
2. $\Omega_I = \{X\}$
3. $\mathcal{M}_I^I(X) = \mathcal{M}_I^O(X) = \{0,1\}$
4. $\mathcal{OD}_I(X) = (\{0,1\}, \{0,1\},$ $\{(I,O) \mid O(t) = 1 - I(t)\})$
5. $\mathcal{OT}_I = (\{0,1\}^2, \{0,1\}^2,$ $\{((I_\perp^T, I_X^T), (O_\perp^T, O_X^T)) \mid O_\perp \equiv 0 \wedge O_X^T = I_X^T\})$

The object and the transmission unit are determined and pre-causal, but for no input there exists a calculation. Hence the canonical I/O-description of $\mathcal{S}_I$ is $\top$. $\mathcal{S}_I$ models an inverter, which output is connected with its input. Unlike in reality the output arrives instantly at the input. If the output of the inverter was delayed a bit, the system would have a calculation for every input and would represent a real I/O-description.

□

Example: Let $\mathcal{T} = \mathbb{N}_0$ and $\mathcal{S}_X = (\mathcal{IM}_X, \mathcal{OM}_X, \Omega_X, \mathcal{M}_X^I, \mathcal{M}_X^O, \mathcal{OD}_X, \mathcal{OT}_X)$ with

1. $\mathcal{IM}_X = \mathcal{OM}_X = \{0,1,2\}$
2. $\Omega = \{X\}$
3. $\mathcal{M}_X^I(X) = \mathcal{M}_X^O(X) = \{0,1,2\}$
4. $\mathcal{OD}_X(X) = (\{0,1,2\}, \{0,1,2\},$ $\{(I,O) \mid (\forall t \in \mathcal{T} \setminus \{1\} : O(t) = I(t)) \wedge$
   $$O(1) = \begin{cases} 0 & : \text{ if } I(1) = 0 \\ 1 & : \text{ if } I(1) = 2 \\ 2 & : \text{ if } I(1) = 1 \end{cases}$$
   $\})$
5. $\mathcal{OT}_X = (\{0,1,2\}^2, \{0,1,2\}^2,$ $\{((I_\perp, I_X), (0,0))\} \cup$ $\{((I_\perp, I_X), (O_\perp, O_X)) \mid O_\perp = I_X \wedge$ $(\forall t \in \mathcal{T} \setminus \{1\} : O_X(t) = I_\perp(t)) \wedge$
   $$O_X(1) = \begin{cases} I_X(1) & : \begin{array}{l} \text{if } I_\perp(0) = 2 \wedge \\ \quad I_\perp(1) = 2 \wedge \\ \quad I_X(1) \neq 0 \end{array} \\ 2 & : \begin{array}{l} \text{if } I_\perp(0) = 2 \wedge \\ \quad I_\perp(1) = 1 \wedge \\ \quad I_X(1) \neq 0 \end{array} \\ 1 & : \text{ otherwise} \end{cases}$$
   $\})$

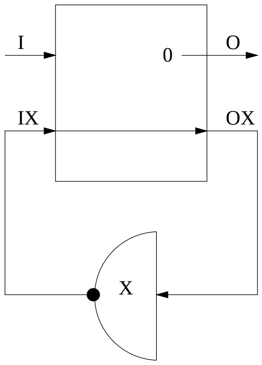

Fig. 3. Inverter

The object and the transmission unit of $\mathcal{S}_X$ are causal.

The canonical I/O-description $\mathcal{D}_{\mathcal{S}_X}$ of $\mathcal{S}_X$ is real. Let $I_1, I_2 \in (\mathcal{T} \to \{0,1,2\})$ with $I_1(0) = I_2(0) = 2$, $I_1(1) = 0$ and $I_2(1) = 2$. Then there exists an $(I_1, O_1) \in \mathcal{D}_{\mathcal{S}_X}$ with $O_1(0) = 2$, but there does not exist any $(I_2, O_2) \in \mathcal{D}_{\mathcal{S}_X}$ with $O_2(0) = 2$. Hence $\mathcal{D}_{\mathcal{S}_X}$ is not pre-causal, though the object and the transmission unit are causal. □

An object based system can be viewed as a mapping from the I/O-descriptions into the I/O-descriptions. This mapping is monotone and hence has a least fixpoint. By this recurrence of object based systems can be defined. We do not know yet, under which conditions the fixpoint has some of the defined properties.

Lemma: Let $\mathcal{S}_1 = (\mathcal{IM}_1, \mathcal{OM}_1, \Omega_1, \mathcal{M}_1^I, \mathcal{M}_1^O, \mathcal{OD}_1, \mathcal{OT}_1)$ and $\mathcal{S}_2 = (\mathcal{IM}_1, \mathcal{OM}_1, \Omega_1, \mathcal{M}_1^I, \mathcal{M}_1^O, \mathcal{OD}_2, \mathcal{OT}_2)$ be object based systems with:
$\forall \omega \in \Omega^T : \mathcal{OD}_1(\omega) \prec \mathcal{OD}_2(\omega)$.
Then the canonical I/O-description of $\mathcal{S}_1$ is more

determined than the canonical I/O-description of $\mathcal{S}_2$. □

Proof: Every calculation of $\mathcal{S}_1$ is a calculation of $\mathcal{S}_2$. □

C. Object Based Systems as Causal I/O-Descriptions

Obviously the canonical I/O-description of an object based system may behave almost arbitrary, even if the objects and the transmission unit are causal. Now a huge class of object based systems with well behaved canonical I/O-descriptions will be identified. It is very important that the canonical I/O-descriptions of these object based systems can be used as objects again.

Definition: Let $\mathcal{S}$ be an object based system.

1. $\mathcal{S}$ is called determined if the objects and the transmission unit are real and determined.
2. $\mathcal{S}$ is called internally delayed if the objects are real and causal and the transmission unit is real and internally delayed with respect to $(\perp, \perp)$.

□

Lemma: Let $\mathcal{S}$ be a determined, internally delayed object based system. Let I be an input and $\mathcal{R}$ be a calculation of $\mathcal{S}$ for input I up to time $t_0 \in \mathcal{T}$.
Then there exists a calculation $\mathcal{R}_1$ of $\mathcal{S}$ for input I that equals $\mathcal{R}$ up to t_0, $\mathcal{R} =_{t_0} \mathcal{R}_1$. □

Proof: In other works [17], [3], [15], [4] Banach's theorem is used. In this setting this technique cannot be used, because it is not possible to continue a calculation by constant time-interval. Thus the constructed function would not be contracting and hence Banach's theorem could not be applied.
Let $\mathcal{S} = (\mathcal{IM}, \mathcal{OM}, \Omega, \mathcal{M}^I, \mathcal{M}^O, \mathcal{OD}, \mathcal{OT})$ be an object based system with $\mathcal{OD}(\omega) = \mathcal{D}_\omega = (\mathcal{I}_\omega, \mathcal{O}_\omega, \mathcal{U}_\omega)$ for $\omega \in \Omega^T$. Let $\mathcal{R} = ((I_\omega, O_\omega)_{\omega \in \hat{\Omega}}, (I_T, O_T))$ be a calculation of $\mathcal{S}$ up to t_0 with $I = O_\perp$.
Since $\mathcal{OT}$ is internally delayed with respect to $(\perp, \perp)$, and there exists a $t_0 < \hat{t} \in \mathcal{T}$, such that for every $\hat{I} \in (\mathcal{T} \to \mathcal{I}_T)$ with $I =_{t_0} \hat{I}$ and $Ext_{\{\perp\}}(I) = Ext_{\{\perp\}}(\hat{I})$, there exists $(\hat{I}, \hat{O}) \in \mathcal{U}_T$ with $O =_{t_0} \hat{O}$ and $Int_{\{\perp\}}(O) =_{\hat{t}} Int_{\{\perp\}}(\hat{O})$.
Then let $O_T = (O^T_\omega)_{\omega \in \hat{\Omega}}$ define the input into the objects up to $\hat{t}$, thus for $\omega \in \Omega$ let $I^1_\omega = O^T_\omega$.
Because the objects are pre-causal there exist $(I^1_\omega, O^1_\omega) \in \mathcal{U}_\omega$ with $O_\omega =_{t_0} O^1_\omega$.
The output of the objects forms the input of the transmission unit up to $\hat{t}$: $I^1_T = (O_\omega)_{\omega \in \hat{\Omega}}$.
Since $\mathcal{OT}$ is internally delayed with respect to $(\perp, \perp)$, there exists $(I^1_T, O^1_T) \in \mathcal{U}_T$ with $O_T =_{t_0} O^1_T$ and $Int_{\{\perp\}}(O_T) =_{\hat{t}} Int_{\{\perp\}}(O^1_T)$. O^1_T defines an output $O^1_\perp$ up to $\hat{t}$.
Thus $\mathcal{R}_1 = ((I^1_\omega, O^1_\omega)_{\omega \in \hat{\Omega}^T})$ is a calculation of $\mathcal{S}$ for I up to $\hat{t}$ which equals $\mathcal{R}$ up to t_0.
By this a sequence of calculations $\mathcal{R}_n = ((I^n_\omega, O^n_\omega))_{\omega \in \hat{\Omega}^T}$ up to t_n can be constructed. Let $\hat{t} = \lim_{n \to \infty} t_n$. Consider $\omega \in \Omega^T$ and $\hat{I}_\omega$ with $I^n_\omega =_{t_n} \hat{I}_\omega$ for all $n \in \mathbb{N}_0$. Since $\mathcal{D}_\omega$ is pre-causal, for every $n \in \mathbb{N}_0$ there exists an $(\hat{I}_\omega, \hat{O}^n_\omega) \in \mathcal{U}_\omega$ with $O^n_\omega =_{t_n} \hat{O}^n_\omega$. Since $\mathcal{D}_\omega$ is determined, this is always the same output: $\forall n, m \in \mathbb{N}_0 : \hat{O}_\omega := \hat{O}^n_\omega = \hat{O}^m_\omega$. Hence $\mathcal{R}_{\hat{t}} = (\hat{I}_\omega, \hat{O}_\omega)_{\omega \in \Omega^T}$ is a calculation up to $\hat{t}$.
If $\hat{t} = \infty$, we are done. If $\hat{t} \neq \infty$, we have to construct a new sequence of calculations. By repeating this (possibly infinite often) we get a calculation of $\mathcal{S}$ for input I. □

This lemma can be used to prove that determined, internally delayed object based systems behave very well.

Lemma: Let $\mathcal{S}$ be a determined, internally delayed object based system with canonical I/O-description $\mathcal{D}_\mathcal{S}$.

1. $\mathcal{D}_\mathcal{S}$ is real.
2. $\mathcal{D}_\mathcal{S}$ is pre-causal.
3. $\mathcal{D}_\mathcal{S}$ is determined.

□

Proof: Let $\mathcal{D}_\mathcal{S} = (\mathcal{I}_\mathcal{S}, \mathcal{O}_\mathcal{S}, \mathcal{U}_\mathcal{S})$ be the canonical I/O-description of $\mathcal{S}$.

1. Obviously, for every input I there exists a calculation $\mathcal{R}_0$ of $\mathcal{S}$ up to 0. Then there exists a calculation $\mathcal{R}_\infty$ of $\mathcal{S}$ for input I. This calcualtion defines an output for input I and hence $\mathcal{D}_\mathcal{S}$ is real.
2. Consider $(I, O) \in \mathcal{U}_\mathcal{S}$, $t \in \mathcal{T}$, and $\hat{I} \in (\mathcal{T} \to \mathcal{I}_\mathcal{S})$ with $I =_t \hat{I}$.
   There exists a calculation $\mathcal{R}_{(I,O)}$ of $\mathcal{S}$ for input I with output O. Since the objects and

the transmission unit are pre-causal, $\mathcal{R}_{(I,O)}$ is a calculation of $\mathcal{S}$ for input $\hat{I}$ with output O up to t. This calculation can be continued to a calculation $\mathcal{R}_{\hat{I}}$ for input $\hat{I}$. $\mathcal{R}_{\hat{I}}$ defines an output that equals O up to t.

3. Let I be an input and $\mathcal{R}_1 = (I^1_\omega)_{\omega \in \Omega^T}$ and $\mathcal{R}_2 = (I^2_\omega)_{\omega \in \Omega^T}$ be calculations of $\mathcal{S}$ for input I. Let $t_0 = \max\{t \mid I^1_T =_t I^2_T\}$.
   Since the transmission unit is determined and pre-delayed, there exists $t_1 > t_0$ with $Int_\perp(O^1_T) =_{t_1} Int_\perp(O^2_T)$. Because the objects are determined and pre-causal, $O^1_\omega =_{t_1} O^2_\omega$ and hence $I^1_T =_{t_1} I^2_T$.
   Therefore $t_0 = \infty$ and $\mathcal{R}_1 = \mathcal{R}_2$.

□

Now we are in the position to present the main result of this paper. The canonical I/O-description of object based systems with causal objects and an internally delayed transmission unit is causal. Hence the canonical I/O-description of such an object based system can itself be used as an object.

Lemma: Let $\mathcal{S}$ be an internally delayed object based system. Then the canonical I/O-description of $\mathcal{S}$ is a real and causal I/O-description. □

Proof: The canonical I/O-description $\mathcal{D}_\mathcal{S} = (\mathcal{I}_\mathcal{S}, \mathcal{O}_\mathcal{S}, \mathcal{U}_\mathcal{S})$ of $\mathcal{S}$ is real due to monotony.

Consider $(I, O) \in \mathcal{U}_\mathcal{S}$. Then there exists a calculation $\mathcal{R}_1 = ((I^1_\omega, O^1_\omega))_{\omega \in \Omega^T}$ of $\mathcal{S}$ for input I with output O. For $\omega \in \Omega^T$ there exists a determined pre-causal I/O-description $\mathcal{D}_{(\omega,1)} = (\mathcal{I}_{(\omega,1)}, \mathcal{O}_{(\omega,1)}, \mathcal{U}_{(\omega,1)})$ with $(I_\omega, O_\omega) \in \mathcal{U}_{(\omega,1)}$. Then the canonical I/O-description $\mathcal{D}_1 = (\mathcal{I}_1, \mathcal{O}_1, \mathcal{U}_1)$ of the object based system consisting of these objects is determined and pre-causal with $(I, O) \in \mathcal{U}_1$. Hence $\mathcal{D}_\mathcal{S}$ is causal. □

VI. Timeless I/O-descriptions

As was already mentioned it is hard to prove for timeless I/O-descriptions that they are causal or even delayed. Therefore a technique for defining causal and delayed I/O-descriptions is needed. Additionally this serves as an example for the use of object based systems.

A. Timeless Objects

A technique for defining a timeless causal I/O-description is:

1. Define a causal time system $\mathcal{D}^P$. One way to ensure that $\mathcal{D}^P$ is causal, is by defining it as a determined pre-causal time system.
2. This causal time system defines a causal canonical I/O-description $\mathcal{D}$.
3. Using $\mathcal{D}$ a causal I/O-description $\mathcal{D}_Z$ is defined that essentially behaves like $\mathcal{D}$. But the received messages need not be processed immediately (but they are processed) and the answers need not be sent immediately (but they are sent). In addition the order of some messages can be exchanged.

Let N_1, N_2 be non-empty sets, $\tau_1 \notin N_1$, $N_1^\tau = N_1 \cup \{\tau_1\}$, $\tau_2 \notin N_2$ and $N_2^\tau = N_2 \cup \{\tau_2\}$. Let $\phi_1 \subset N_1 \times N_1$ and $\phi_2 \subset N_2 \times N_2$.

Then $\mathcal{D}_T = (N_1^\tau \times N_2^\tau, N_2^\tau \times N_1^\tau,$
$\{((I_\perp, I_X), (O_\perp, O_X)) \mid (I_X, O_\perp) \in \mathcal{D}_{Buf(N_1^\tau, \phi_1)}$
$\wedge\, (I_\perp, O_X) \in \mathcal{D}_{Buf(N_2^\tau, \phi_2)}\})$ is internally delayed with respect to $(\perp, \perp)$.

Let $\mathcal{D} = (N_1^\tau, N_2^\tau)$ be an I/O-description, then
$\mathcal{BUF}^{(\phi_2,\tau_2)}_{(\phi_1,\tau_1)}(\mathcal{D})$ is the canonical I/O-description of the object based system $(N_1^\tau, N_2^\tau, \{X\}, \{(X, N_1^\tau)\}, \{(X, N_2^\tau\}, \{(X, \mathcal{D})\}, \mathcal{D}_T)$. The transmission unit gets messages from the outside. It can delay these messages and swap according to ϕ_1. But it sends all received messages to the object behaving according $\mathcal{D}$. The transmission union gets the output of the objects. It can delay this output and can swap messages according to ϕ_2. Again it sends all received messages to the outside.

If $\mathcal{D}$ is causal, then $\mathcal{BUF}^{(\phi_2,\tau_2)}_{(\phi_1,\tau_1)}(\mathcal{D})$ is causal.

B. Timeless transmission units

A similar technique for defining timeless transmission units is a bit more complicated, because the input of a transmission unit is a tuple of inputs. Each component can be delayed independently and the transmission unit must be delayed. A transmission unit can get many messages at the same time that all must be sent to one object. If the transmission unit shall be described by a determined I/O-description, this problem must be handled properly.

A delayed transmission unit can be defined as follows:

1. Define a causal time system $\mathcal{D}^P$. $\mathcal{D}^P$ does not send tuples of usual messages but tuples of finite sets of messages. One way to ensure that $\mathcal{D}^P$ is causal, is by defining it as a determined pre-causal time system.
2. $\mathcal{D}^P$ defines a causal canonical I/O-description $\mathcal{D}$.
3. $\mathcal{D}$ is changed into a causal I/O-description $\mathcal{D}_1$, that does not send tuples of finite sets of messages any more but tuples of the messages contained in the sets.
4. $\mathcal{D}_1$ is used to define a causal I/O-description $\mathcal{D}_2$ that behaves like $\mathcal{D}_1$, except that the components of the messages are delayed independently.
5. $\mathcal{D}_2$ is used to define a delayed I/O-description $\mathcal{D}_3$ that behaves like $\mathcal{D}_2$, except that the answers are slightly delayed.

B.1 Separation of Messages

Let N be a non-empty set, $\tau \in N$ and $\mathcal{D}^P$ the time system $(\mathcal{P}^E(N), N, \{(I,O) \mid (\forall t \in \mathcal{T} : |\{t_0 \in \mathcal{T} \mid t_0 < t \wedge I(t_0) \neq \emptyset\}| < \infty) \wedge \exists f : \mathcal{MT}_\tau(O) \rightarrow \mathcal{MT}_\emptyset(I):$

1. f is monoton increasing [messages do not overtake]
2. $\forall \hat{t} \in \mathcal{MT}_\tau(O) : \hat{t} \geq f(\hat{t})$ [$\mathcal{D}^P$ is pre-causal]
3. $\forall t_0, t_1 \in \mathcal{MT}_\tau(O) : (t_0 \neq t_1 \wedge f(t_0) = f(t_1) \Rightarrow O(t_0) \neq O(t_1))$ [an incoming message is not sent more than one time]
4. $\forall \hat{t} \in \mathcal{MT}_\emptyset(I) : O(f^{-1}(\hat{t}) = I(t) \setminus \{\tau\}$ [all incoming messages are sent]

$\})$. $\mathcal{D}^P$ is causal. Let $\mathcal{D}_{\mathcal{AUS}(N,\tau)}$ be the canonical I/O-description of $\mathcal{D}^P$, then $\mathcal{D}_{\mathcal{AUS}(N,\tau)}$ is causal.

Let M be a set and for $m \in M$ let N_m be a set with $\tau_m \in N_m$. Then let $\mathcal{D}_{\mathcal{IAUS}((N_m,\tau_m)_{m\in M})}$ be the canonical I/O-description of the object based system
$(\prod_{m\in M} \mathcal{P}^E(N_m), \prod_{m\in M} N_m, \{X_m \mid m \in M\},$
$\{(X_m, \mathcal{P}^E(N_m)) \mid m \in M\},$
$\{(X_m, N_m) \mid m \in M\},$
$\{(X_m, \mathcal{D}_{\mathcal{AUS}(N_m,\tau_m)}) \mid m \in M\},$
$\{((((I_\perp^m)_{m\in M}), (I_m)_{m\in M}), (((O_\perp^m)_{m\in M}),$
$(O_m)_{m\in M})) \mid \forall m \in M : O_\perp^m = I_m \wedge O_m = I_\perp^m\})$.
This object based system gets tuples of finite sets of messages as input. Every component of a tuple is sent to a corresponding object that separates the messages contained in the finite sets. The outputs of the objects are put together to a tuple of messages and sent to the outside.

$\mathcal{D}_{\mathcal{IAUS}((N_m,\tau_m)_{m\in M})}$ is causal.

Let $\mathcal{D} = (\mathcal{I}, \prod_{m\in M} \mathcal{P}^E(N_m), \mathcal{U})$ be an I/O-description. Then $\mathcal{AU}_{(\tau_m)_{m\in M}}(\mathcal{D})$ is the canonical I/O-description of the object based system
$(\mathcal{I}, \prod_{m\in M} N_m, \{X\}, \{(X,\mathcal{I})\},$
$\{(X, \prod_{m\in M} \mathcal{P}^E(N_m))\}, \{(X,\mathcal{D})\},$
$\{((I_\perp, I_X), (O_\perp, O_X)) \mid O_X = I_\perp \wedge (I_X, O_\perp) \in \mathcal{D}_{\mathcal{IAUS}((N_m,\tau_m)_{m\in M})}\})$
The object based system gets messages that are sent to the object immediately. The object sends tuples of finite sets of messages. The transmission unit separates the messages contained in the finite sets and sends them to the outside.

If $\mathcal{D}$ is causal, then $\mathcal{AU}_{(\tau_m)_{m\in M}}(\mathcal{D})$ is causal.

B.2 Independent Delay and Swapping

Let M be a set. For $m \in M$ let N_m be a set and $\tau_m \notin N_m$, $N_m^\tau = N_m \cup \{\tau_m\}$. Let $\phi_m \subset N_m \times N_m$. Then $\mathcal{D}_{\mathcal{IBUF}((N_m^\tau,\phi_m)_{m\in M})}$ is the canonical I/O-description of the object based system
$(\prod_{m\in M} N_m^\tau, \prod_{m\in M} N_m^\tau, \{X_m \mid m \in M\},$
$\{(X_m, N_m^\tau) \mid m \in M\},$
$\{(X_m, N_m^\tau) \mid m \in M\},$
$\{(X_m, \mathcal{D}_{Buf(N_m^\tau,\phi_m)}) \mid m \in M\},$
$\{(((I_\perp^m)_{m\in M}), (I_m)_{m\in M}), (((O_\perp^m)_{m\in M}),$
$(O_m)_{m\in M}) \mid O_m = I_\perp^m \wedge O_\perp^m = I_m\})$.
This object based system gets tuples of messages as input. Each component is sent to an independent buffer. The output of the buffers is put together and sent to the outside.

$\mathcal{D}_{\mathcal{IBUF}((N_m^\tau,\phi_m)_{m\in M})}$ is causal.

Let $\mathcal{D} = (\mathcal{I}, \prod_{m\in M} N_m, \mathcal{U})$ be an I/O-description.

Then $\mathcal{IBUF}_{(\phi_m,\tau_m)_{m\in M}}(\mathcal{D})$ is the canonical I/O-description of the object based system
$(\mathcal{I}, \prod_{m\in M} N_m, \{X\}, \{(X,\mathcal{I})\}, \{(X, \prod_{m\in M} N_m)\},$
$\{(X,\mathcal{D})\}, \{((I_\perp, I_X), (O_\perp, O_X)) \mid O_X = I_\perp \wedge$
$(I_X, O_\perp) \in \mathcal{D}_{\mathcal{IBUF}((N_m^\tau,\phi_m)_{m\in M})}\})$.
The object based system gets messages that are sent to the object immediately. The object answers with tuples of messages. Each component of the tuples is delayed individually and then sent to the outside. If $\mathcal{D}$ is causal, then $\mathcal{IBUF}_{(\phi_m,\tau_m)_{m\in M}}(\mathcal{D})$ is causal.

B.3 Delay of Messages

Let $\mathcal{D} = (\mathcal{I}, \mathcal{O}, \mathcal{U})$ be an I/O-description with $\tau \in \mathcal{O}$. Then $\mathcal{VER}_\tau(\mathcal{D})$ is the canonical I/O-description of the object based system
$(\mathcal{I}, \mathcal{O}, \{X\}, \{(X,\mathcal{I})\}, \{(X,\mathcal{O})\}, \{(X,\mathcal{D})\}, \mathcal{D}_T)$
with $\mathcal{D}_T = (\mathcal{I} \times \mathcal{O}, \mathcal{O} \times \mathcal{I},$
$\bigcup_{0\neq t\in\mathcal{T}} \{((I_\perp, I_X), (O_\perp, O_X)) \mid$

1. $O_X = I_\perp$ [input from the outside is sent to the object immediately]
2. $\forall t_0 \in \mathcal{T} : O_\perp(t_0 + t) = I_X$ [the output of the object is output to the outside with delay t
3. $\forall t_0 < t : O_\perp(t_0) = \tau$

}).
This object based system sends the input from the outside immediately to the object. The output of the object is slightly delayed and then sent to the outside.

If $\mathcal{D}$ is causal, then $\mathcal{VER}_\tau(\mathcal{D})$ is delayed.

B.4 Composition

Let M be a set. For $m \in M$ let N_m be a set, $\tau_m \notin N_m$, $N_m^\tau = N_m \cup \{\tau_m\}$ and $\phi_m \subset N_m \times N_m$. Let $\mathcal{D} = (\mathcal{I}, \prod_{m \in M} \mathcal{P}^E(N_m^\tau), \mathcal{U})$ be an I/O-description. Then $\mathcal{UF}_{(\phi_m,\tau_m)_{m\in M}}(\mathcal{D}) = \mathcal{VER}_{(\tau_m)_{m\in M}}(\mathcal{IBUF}_{(\phi_m,\tau_m)_{m\in M}}(\mathcal{AU}_{(\tau_m)_{m\in M}}(\mathcal{D})))$

If $\mathcal{D}$ is causal, then $\mathcal{UF}_{(\phi_m,\tau_m)_{m\in M}}(\mathcal{D})$ is delayed.

VII. Example

In this part an example is presented that illustrates the use of object based systems and the definition of timeless objects and a timeless transmission unit. Moreover this illustrates a technique for 'creating' new objects in this model.

For this purpose an object based system $\mathcal{S} = (\mathcal{IM}, \mathcal{OM}, \Omega, \mathcal{M}^I, \mathcal{M}^O, \mathcal{OD}, \mathcal{OT})$ is introduced, which implements a stack (LIFO) for elements of $\mathcal{V}$. The system consists of one object (with address H) that coordinates the stack and infinite many objects (with addresses in $\mathbb{N}_0$) that can store one value of $\mathcal{V}$ and the address of one object. The coordinating object stores the address of the top element of the stack. Every element of the stack stores the value it represents and the address of the object representing the next value below. The bottom element of the stack stores the address H of the coordinating object. If a new element is pushed onto the stack, the coordinating object introduces an object, not yet engaged in the stack, by initializing it with the value to be stored and the address of the object representing the old top element of the stack.

A more comprehensive example can be found in [12].

A. Messages

1. The object based system can get the empty message τ, the pop message $Pop()$, and a push message $Push(val)$ from the outside.
$\mathcal{IM} = \{\tau, Pop()\} \cup \{Push(val) \mid val \in \mathcal{V}\}$.
It can send the empty message τ, and a result message $Result()$ indicating an error or $Result(val)$ returning the value val to the outside.
$\mathcal{OM} = \{\tau, Result()\} \cup \{Result(val) \mid val \in \mathcal{V}\}$.
2. The coordinating object (with address H) can get the empty message τ, the pop message $Pop()$, a push message $Push(val)$, and a content message $Content(val, addr)$.
$\mathcal{M}^I(H) = \{\tau, Pop()\} \cup \{Push(val), Content(val, addr) \mid val \in \mathcal{V}, addr \in \mathbb{N}_0 \cup \{H\}\}$.
It can send the empty message τ, messages to initialize other objects $Init(rec, val, addr)$, to query the content of an object $Query(rec)$, and to answer to the outside $Result()$ and $Result(val)$.
$\mathcal{M}^O(H) = \{\tau, Result()\} \cup \{Query(rec), Init(rec, val, addr), Result(val) \mid rec \in \mathbb{N}_0, val \in \mathcal{V}, addr \in \mathbb{N}_0 \cup \{H\}\}$.
3. The other objects (with address $rec \in \mathbb{N}_0$) can get the empty message τ, the query message $Query()$ and the initialization message $Init(val, addr)$.
$\mathcal{M}^I(rec) = \{\tau, Query()\} \cup \{Init(val, addr) \mid val \in \mathcal{V}, addr \in \mathbb{N}_0 \cup \{H\}\}$.
They can send the empty message τ and a content message $Content(val, addr)$ to tell the coordinating object their contents.
$\mathcal{M}^O(rec) = \{\tau\} \cup \{Content(val, addr) \mid val \in \mathcal{V}, addr \in \mathbb{N}_0 \cup \{H\}\}$
4. By the definitions above the messages received and sent by the transmission unit are determined.

B. Objects

B.1 The Coordinating Object

First a determined I/O-description $\mathcal{D}_{(H,d)}$ is defined by an implementation Imp_H which possesses a state $(top, next)$. top is the address of the object representing the top of the stack. All objects with addresses greater or equal $next$ have never been used before. At time $t = 0$ the state of the coordinating object is $(H, 0)$.

1. If Imp_H receives the empty message τ it sends the empty message τ and does not

change its state.

2. If Imp_H is in state $(top, next)$ and receives the message $Pop()$, it does not change its state and sends $Query(top)$, if $top \neq H$. Otherwise it does not change its state and sends $Result()$.
3. If Imp_H is in state $(top, next)$ and receives the message $Push(val)$, it changes to the state $(next, next+1)$ and sends the message $Init(next, val, top)$.
4. If Imp_H is in state $(top, next)$ and receives the message $Content(val, addr)$, it changes to the state $(addr, next)$ and sends the message $Result(val)$.

The coordinating object behaves like $\mathcal{D}_{(H,d)}$ except that the answers need not be sent immediately.
$\mathcal{OD}(H) = \mathcal{BUF}^{(\emptyset,\tau)}_{(\emptyset,\tau)}(\mathcal{D}_{(H,d)})$.

B.2 Other Objects

First a determined I/O-description $\mathcal{D}_{(\mathbb{N}_0,d)}$ is defined by an implementation $Imp_{\mathbb{N}_0}$ which possesses a state $(val, addr)$. val is the value represented by this object and $addr$ is the address of the object, representing the next value on the stack. At time $t = 0$ the state of these objects is arbitrary.

1. If $Imp_{\mathbb{N}_0}$ receives an empty message τ, it does not change its state and sends the empty message τ.
2. If $Imp_{\mathbb{N}_0}$ is in state $(val, addr)$ and receives the query message $Query()$, it does not change its state and sends the message $Content(val, addr)$.
3. If $Imp_{\mathbb{N}_0}$ is in state $(val, addr)$ and receives the initialization message $Init(val1, addr1)$, it changes its state to $(val1, addr1)$ and sends the empty message τ.

The other objects behave like $\mathcal{D}_{(\mathbb{N}_0,d)}$ except that the answers need not be sent immediately.
$\mathcal{OD}(n) = \mathcal{BUF}^{(\emptyset,\tau)}_{(\emptyset,\tau)}(\mathcal{D}_{(n,d)})$ for $n \in \mathbb{N}_0$.

C. The Transmission Unit

First a determined time system $\mathcal{D}^P_{(T,d)} = (\mathcal{I}_{(T,d)}, \mathcal{O}_{(T,d)}, \mathcal{U}^P_{(T,d)})$ is defined. An output is only defined for those inputs $I = (I_n)_{n \in \mathbb{N}_0 \cup \{H,\perp\}}$ with both:

1. finite many messages unequal $(\tau)_{n \in \mathbb{N}_0 \cup \{H,\perp\}}$ for any finite time-interval: $\forall t \in \mathcal{T} : |\{\hat{t} \in \mathcal{T} \mid \hat{t} \leq t \wedge I(\hat{t}) \neq (\tau)_{n \in \mathbb{N}_0 \cup \{H,\perp\}}\}| < \infty$
2. and finite many components are unequal τ for all times:
$\forall t \in \mathcal{T} :$
$|\{n \in \mathbb{N}_0 \cup \{H,\perp\} \mid I_n(t) \neq \tau\}| < \infty$

For these inputs, $(\mathcal{I}_{(T,d)}, \mathcal{O}_{(T,d)}, \mathcal{U}^P_{(T,d)})$ is defined as follows:
$((I_n)_{n \in \mathbb{N}_0 \cup \{H,\perp\}}, (O_n)_{n \in \mathbb{N}_0 \cup \{H,\perp\}}) \in \mathcal{U}^P_{(T,d)} \Leftrightarrow$
$\forall t \in \mathcal{T} :$

1. $\forall rec \in \mathbb{N}_0 : O_{rec}(t) = \{Query() \mid I_H(t) = Query(rec)\} \cup \{Init(val, addr) \mid I_H(t) = Init(rec, val, addr)\}$
2. $O_\perp(t) = \{Result() \mid I_H(t) = Result()\} \cup \{Result(val) \mid I_H(t) = Result(val)\}$
3. $O_H(t) = \{Pop() \mid I_\perp(t) = Pop()\} \cup \{Push(val) \mid I_\perp(t) = Push(val)\} \cup \{Content(val, addr) \mid \exists n \in \mathbb{N}_0 : I_n(t) = Content(val, addr)\}$

This time system $\mathcal{D}^P_{(T,d)}$ defines a causal I/O-description $\mathcal{D}^P_{(T,c)}$.
The I/O-description describing the transmission unit is then $\mathcal{UF}_{(\emptyset,\tau)_{n \in \mathbb{N}_0 \cup \{H,\perp\}}}(\mathcal{D}_{(T,c)})$, which is internally delayed.

D. Analysis

It is obvious that this object based system represents a stack. But this implementation can get confused, if it gets a non empty message, before it has answered the last pop message. Hence this is only a sequential stack. The definition of this object based system is relatively easy, though it already possesses some complexity. A definition of a parallel stack is a bit more complicated but should be manageable, too.

VIII. Conclusion

A really concurrent model has been introduced that can be used as a model for all – timed and untimed – systems respecting encapsulation. Models like the actor model or the layered semantics of POOL can be modeled easily. The model is fully abstract in the sense that an object exactly represents the behavior observable by the outside – i.e. by contexts. By modeling the actor model or the layered semantics of POOL with this model, these concepts become hierarchical and can easily be used in other contexts.

References

[1] Gul Agha, Ian A. Mason, Scott Smith, and Carolyn Talcott. Towards a theory of actor compu-

tation. In W. R. Cleaveland, editor, *CONCUR '92, Third International Conference on Concurrency Theory*, volume 630 of *Lecture Notes in Computer Science*, pages 565–579, Berlin, August 1992. Springer-Verlag.

[2] Gul A. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* The MIT Press series in artificial inteligence. MIT Press, 1986.

[3] Pierre America and Jan Rutten. A layered semantics for a prallel object-oriented language. In J.W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages*, volume 489 of *Lecture Notes in Computer Science*, pages 91–123, Berlin, May 1990. REX School, Springer-Verlag.

[4] J. W. de Bakker and J. I. Zucker. Processes and the denotational semantics of concurrency. In j. W. de Bakker and J. van Leeuwen, editors, *Foundations of Computer Science IV, Distributed Systems: Part 2, Semantics and Logic*, volume 159 of *Mathematical Centre Tracts*, pages 45–100. Mathematisch Centrum, Amsterdam, Niederlande, 1983.

[5] Joost Engelfriet, Georg Leih, and Grzegorz Rozenberg. Net-based description of parallel object-based systems, or pots and pops. In J.W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages*, volume 489 of *Lecture Notes in Computer Science*, pages 229–273, Berlin, May 1990. REX School, Springer-Verlag.

[6] C. A. R. Hoare. *Communicating Sequential Processes.* Series in Computer Science. Prentice Hall, 1985.

[7] Kohei Honda and Mario Tokoro. On asynchronous communication semantics. In M. Tokoro, O. Nierstasz, and P. Wegner, editors, *Object-Based Concurrent Computing, ECOOP '91 Workshop*, volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer-Verlag, July 1991.

[8] Bengt Jonsson. A fully abstract trace model for dataflow networks. In *Sixteenth Annual ACM Symposium on Principles of Programming Languages*, pages 155–165. ACM Press, 1989.

[9] M. D. Mesarović, D. Macko, and Y. Takahara. *Theory of Hierarchical, Multilevel, Systems*, volume 68 of *Mathematics in Science and Engeneering.* Academic Press, 1970.

[10] M.D. Mesarović and Y. Takahara. *Abstract Systems Theory*, volume 116 of *Lecture Notes in Control and Information Science.* Springer-Verlag, 1989.

[11] Robin Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 66 Wood Labe End, Hemel Hempstead Hertfordshire, HP2 4RG, 1989.

[12] Holger Naundorf. *Ein allgemeines Modell für objektbasierte Systeme (to appear).* Phd thesis, Universität Paderborn, Germany, 1995.

[13] Nielson and Engberg. A calculus of communicating systems with label passing. Research Report DAIMI PB-208, Computer Science Department, University of Aarhus, 1986.

[14] James R. Russel. On oraclizable networks and kahn's principle. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 320–328, January 1990.

[15] Etsuya Shibayama. Sematic layers of object-based concurrent computing. In M. Tokoro, O. Nierstasz, and P. Wegner, editors, *Object-Based Concurrent Computing, ECOOP '91 Workshop*, volume 612 of *Lecture Notes in Computer Science*, pages 99–118. Springer-Verlag, July 1991.

[16] Mario Tokoro and Kazunori Takashio. Toward languages and formal systems for distributed computing. In Rachid Guerraoui, Oscar M. Nierstrasz, and Michel Riveill, editors, *ECOOP '93 – Object-Based Distributed Programming*, volume 791 of *Lecture Notes in Computer Science*, pages 93–110, Berlin, July 1993. Springer-Verlag.

[17] Robert Kim Yates. Networks of real-time processes. In Eike Best, editor, *CONCUR '93, 4th International Conference on Concurrency Theory*, volume 715 of *Lecture Notes in Computer Science*, pages 384–397, Berlin, August 1993. Springer-Verlag.