DEVELOPING SYSTEM DYNAMICS MODELS FROM CAUSAL LOOP DIAGRAMS

THOMAS BINDER, ANDREAS VOX, SALIM BELYAZID, HÖRDUR HARALDSSON, AND MATS SVENSSON

ABSTRACT. In this paper we discuss how a Causal Loop diagram (CLD) can be labeled and structured incrementally in order to finally transform it into a Stock and Flow diagram. A CLD does not contain enough information to describe a model uniquely. Hence the decisions on how to transform the CLD cannot be made automatically; they must be based on information about the modeled system. We describe a general set of possible transformation steps and offer guidance on when to choose which step. Some suggesting simplifications of the general setting will be discussed and illustrated by an example. The main application area of the described interactive algorithm is software development. However, it might also give suggestions on the reorganization of the current system dynamics workflows.

1. INTRODUCTION

Causal Loop diagrams (CLDs) have long been used in standard system dynamics practice for purposes connected with simulation modeling. They are nowadays mostly used prior to simulation analysis, to depict the basic causal mechanisms hypothesized to underlie the reference mode of behavior over time, that is, for articulation of a dynamic hypothesis of the system as endogenous consequences of the feedback structure ([Randers, 1980], [Richardson, 1999], [Sterman, 2000]). It also forms a connection between structure and decisions that generate system behavior. Later, CLDs have started to be used for purposes not necessarily related to model building, namely, for detailed system description and for stand-alone policy analysis ([Wolstenholme, 1999], [Homer & Oliva, 2001]).

The other common notation for system dynamics and system thinking are Stock-and-Flow diagrams (SFDs). Proponents of CLDs laude their accessibility to non-experts and claim that SFDs are useful only for people who understand how they work. Proponents of SFDs criticize the ambiguity and lack of detail in CLDs which prevents simulation of the modeled systems and prefer at least to start with stocks first [Ford, 1999]. Haraldsson [Haraldsson, 2004] and others propose to use CLDs for brainstorming and then to switch to an SFD which models the system exactly. This at once raises the question how CLDs can be used as a base for an SFD.

CLDs can be a good start for system modeling [Haraldsson & Sverdrup, 2003]. However, the transition to SFDs is not straightforward. The information on SFDs is hidden

in the CLDs, collapsed into links and factors. Extracting stocks, flows and auxiliaries from the CLDs requires further investigation of the links and what they represent. This process may increase the number of factors in the system. In order to develop the CLD further, the modeler is therefore required to have in-depth knowledge about the system considered.

We propose a process which consists of four phases:

(i) The modeler labels the initial CLD in an appropriate way. This means deciding which factors are stocks, and which are flows or auxiliaries, and specifying which links represent flow dependencies and which information dependencies. The resulting *labeled CLD* usually reveals inconsistencies in the initial CLD, such as missing factors or illicit links.
(ii) The modeler then uses manually controlled steps to incrementally transform the labeled CLD. These steps are guided by a handful of constraints which characterize syntactic inconsistencies of the model. For each inconsistency the modeler has to decide on a specific transformation step which best fits the intended meaning of the model. The result of this phase, a labeled CLD which fulfills the constraints mentioned above, will be called a *structured CLD*.
(iii) It is then possible to transform the structured CLD automatically into an SFD.
(iv) The modeler quantifies the SFD, i.e. provides parameters, initial values and formulas for information dependencies. This yields a quantified *system dynamics model* (or *model* for short), which can be simulated.

Alternatively one can avoid using SFDs altogether: since the structured CLD and the SFD are equivalent, the modeler can also skip Step (iii) above and quantify the structured CLD instead of the SFD (Step (iv)).

The constraints we propose for a structured CLD must be sufficient to ensure a structure which can be converted into an SFD, and it should be easy to detect their violation. Most of the constraints relate only to a local part of the whole CLD.

The set of transformations for Phase (ii) must also meet certain requirements. On the one hand, each transformation should eliminate inconsistencies without introducing (if possible) new ones. On the other hand, the available steps must be sufficient to convert any labeled CLD into a structured CLD.

In reality, though, the modeler will mix these transformations with normal editing operations. Due to new insights from reviewing inconsistencies, she/he will often have to undo some steps already taken, introduce new factors and links, or eliminate others. In fact, these are exactly the same considerations one would make when developing an SFD from scratch. The advantage of our approach is that one can start with a CLD and that the steps show a path how to incrementally refine it into a SFD resp. structured CLD.

Formal descriptions of relationships between CLDs and SFDs have been given by Burns [Burns, 2001]; the main difference to the work presented here is the dynamic

change of the CLD into an SFD, and on the other hand, a link-centered way of looking at these diagrams, contrasting the factor-centered way of [Burns, 2001]. Burns and Ulgen also study a component-centered view of SFDs, cf. [Burns & Ulgen, 2002a] and [Burns & Ulgen, 2002b].

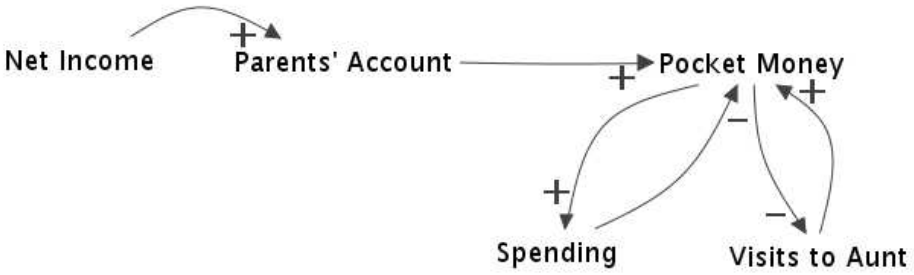
This paper is organized as follows. In Section 2 we give an introductory example. In Section 3 we will recapture the notions of CLD and SFD. The subsequent three sections give a detailed description of the steps (i), (ii), (iii) above. Section 7 describes variants of the described interactive algorithm and ideas on how it can be used in a software tool. We finish with another example (Section 8) and a summary (Section 9).

We assume a basic familiarity with the notions of sets and graphs.

2. An Example: Pocket Money

As an example we will develop a simple model of a child receiving pocket money from its parents. We start with the CLD given in Figure 1. On the one hand, the more money the parents earn the more they are likely to give to their child[1]. On the other hand, the CLD has two feedback loops: The first describes the highly probable fact that spending of the child increases with the available amount of pocket money, and spending decreases this amount. The other feedback loop describes the observation that the aunt hands over money to the child whenever it comes to visit. However, the child does not like its aunt too much, so with increasing budget it is less inclined to see her.

FIGURE 1. The initial CLD of the pocket money model.



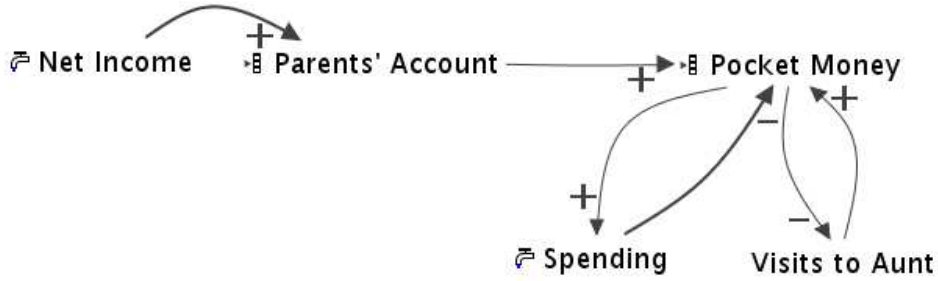
As a first step we have to label this CLD, which means specifying the stocks first. Obviously, the stocks are the *Parent's account* and *Pocket Money*, which is the child's account.

In the next step, we look at all links running into the chosen stocks. For each such link we must decide whether it represents a material flow. If this is the case, the link type

[1]The sole purpose of this model is to illustrate the structural aspects discussed. Therefore, we will not comment on any pedagogical aspects.

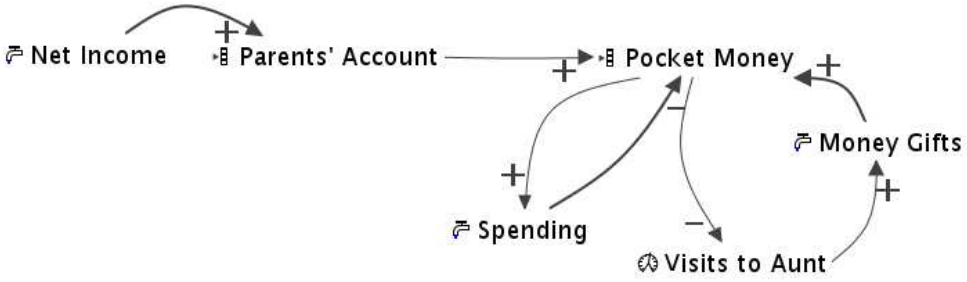
is changed and the type of the factor at the beginning of the link is changed to flow. Figure 2 shows the labeled CLD resulting from these operations.

FIGURE 2. The labeled CLD of the pocket money model. Stocks and flows are marked by meter and tap icons, respectively. Flow dependencies are drawn as thick lines.



We will now transform the labeled CLD into a structured CLD. There is one link running from *Visits to Aunt* to *Pocket Money* which remained untouched, since it is not a material flow. To fix this, we add a new flow factor named *Money Gifts* (Transformation IF, see Section 5.2). Figure 3 shows the resulting diagram.

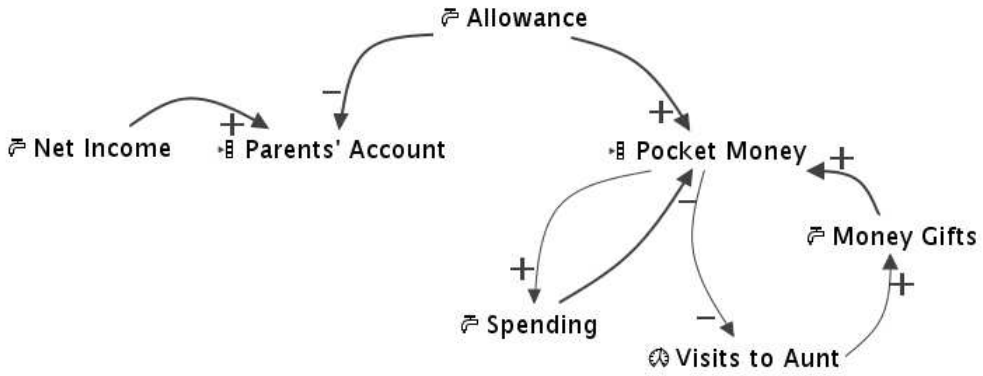
FIGURE 3. The remaining information dependency of the stock *PocketMoney* is treated by applying Transformation IF. Auxiliary factors are marked by the clock icon.



In Figure 3 there is only one violation of the conditions of a structured CLD, namely the link running between the two stocks. We discover that this is a conservative material flow and add a new flow factor *Allowance* (Transformation IMF, see Section 5.3). Figure 4 shows the state after this transformation.

We now have a structured CLD. However, it is not satisfactory, since the new factor *Allowance* is an input factor. This observation makes us realize that there is more to

FIGURE 4. Inserting a flow yields a structured CLD. However, this is not complete since the new flow is an input factor.



do. We extend the model as shown in Figure 5. In this particular example, the diagram remains a structured CLD after the extension; this may not always be the case.

Assume we are content with this model for now. Due to the structured CLD property, the diagram can be switched back and forth between the CLD and SFD views. The switching does not require any more user interaction. Figure 6 shows the SFD corresponding to the structured CLD in Figure 5.

FIGURE 5. The structured CLD of the model. The inserted flow has been connected to the model by adding one auxiliary factor and three information links.

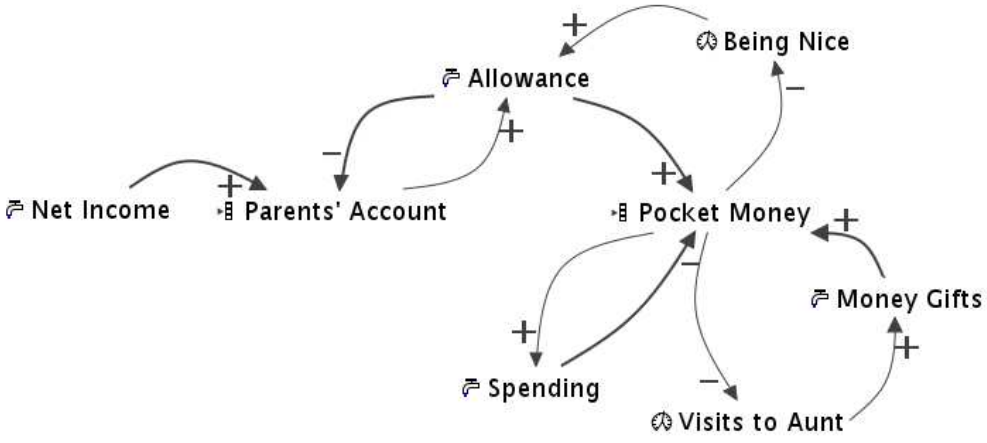
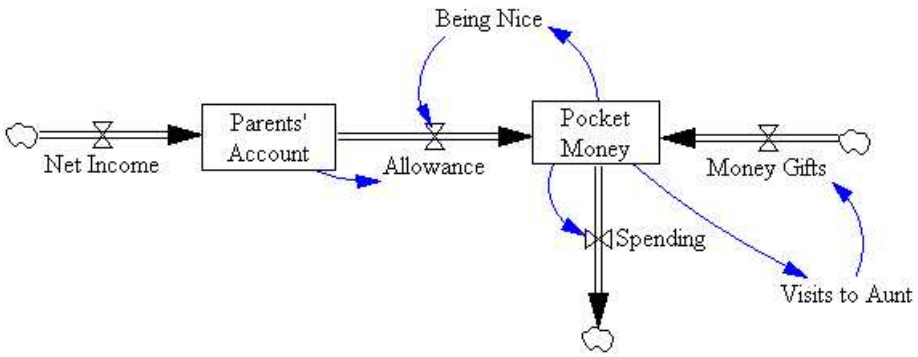


FIGURE 6. The SFD of the pocket money model.



3. CLDs AND SFDs

Both CLDs and SFDs are used to describe complex systems. In this section we will provide a short description, give a definition of both notations as graphs and mention what information is lacking for a complete SD model.

3.1. **Causal Loop Diagrams (CLDs).**

Causal Loop diagrams are used to document relevant factors and the causal relationships between them.

CLDs consist of factors and links connecting the factors. Any link has annotations about its polarity and delay. The polarity tells whether a dependency has positive polarity (if the cause increases, the effect will also increase compared with the situation where the cause did not change) or negative polarity (if the cause increases, the effect will decrease compared with the situation where the cause did not change). The definition of polarity is subtle (cf. [Richardson, 1995]), since even with positive polarity there are scenarios where the cause goes down while the effect goes up (even when there are no other causes for the same effect).

Definition. A *Causal Loop diagram (CLD)* is a tuple $(V; E; \varphi; D)$, where the pair (V, E) is a directed graph, with V and $E \subseteq V \times V$ denoting the set of factors and the set of links between them, respectively. We only consider connected (that is, connected as an undirected graph) CLDs.

Hence, any link $e = (v_1, v_2) \in E$ is an ordered pair of two factors v_1, v_2 indicating that e starts at v_1 and ends at v_2. We assume that the graph has no circles of length one and no duplicate links (pointing in the same direction).

Moreover, there is a function $\varphi : E \to \{\pm 1\}$ called the *polarity map*, i.e. all arrows are labeled as either $+$ or $-$. All *links with delay* of a CLD are collected into the set $D \subseteq E$.

A *feedback loop* is a directed circle of the CLD graph.

3.2. **Stock-and-Flow Diagrams (SFDs).** SFDs are composed of factors whose type is exactly one of *stock*, *flow*, *auxiliary*, or *system boundary*. The system boundary is a special stock which represents an anonymous source or sink. There are two types of links: material flows, which are roughly in a one-to-one correspondence with flow factors, and information dependencies. Material flows may only connect stocks; information dependencies must not point to stocks.

Definition. Consider a set of factors V and a partition of V into three subsets $S \neq \emptyset$ (for stocks), F (for flows) and A (for auxiliaries). The element $\text{☁} \in S$ is a special stock used to denote the system boundary.

A *Stock-and-Flow diagram (SFD)* is a tuple $(V, S, F; m, I; D)$. Here $m : F \to S \times S$ describes source and target of the material flows. The function m

(i) takes any value of S at least once in either component and
(ii) does not take values in $\{(s, s) \mid s \in S\}$.

Moreover, $I \subseteq V \times (F \cup A)$ describes the information dependencies. I is a binary relation such that the graph (V, I) does not contain any directed circles.

Finally, $D \subseteq I$ denotes the set of all information dependencies with delay.

We use the following notation for SFDs:

stock	$\boxed{s}$,	system boundary	☁,
flow	f,	auxiliary	a,
material flow	$\overset{f}{\Longrightarrow}$,	information dependency	$\longrightarrow$.

Note that we do not distinguish flows and auxiliaries notationally. Instead, flows are located next to their corresponding material flow. Also, in contrast to the work of Burns et al., there is no distinction between input, output and parameter factors.

3.3. **Quantification.** The process of turning an SFD into a model is called *quantification*, which will be briefly discussed here. In order to quantify an SFD, the modeler has to provide

(i) Start and end time for the simulation run,
(ii) Formulas for all flow and auxiliary factors, which includes the specification of the delay values or functions for all links in D,
(iii) Values or time-dependent functions (defined between start and end time from (i)) for all input factors (i.e. factors without incoming dependencies),
(iv) Initial values for all stocks, and
(v) Initial values or time-dependent functions for all factors which have outgoing dependencies with delays.

The data in (iv) and (v) is often simply assumed to be zero. Another technique to avoid having to specify (v) is to simulate the model until it may reach a steady state, in which case one simply discards the initial time segment of the simulation.

Usually one also expects dimensional consistency for all values and formulas.

4. From CLDs to labeled CLDs

To produce a labeled CLD, the modeler must identify the stocks and flows among the factors and also the types of links. This is a crucial step on the way to a model.

More precisely, the modeler has to find partitions of the factors and dependencies which match the following definition.

Definition. A partition of V into three subsets $S \neq \emptyset$ (for stocks), F (for flows) and A (for auxiliaries) on a CLD $(V; E; \varphi; D)$ is called a *labeling of the factors*.

A partition of E into two subsets M (for *flow dependencies*) and I (for *information dependencies*) is called a *labeling of the links*.

The *loop condition* is said to be satisfied if along each feedback loop there is at least one stock or one flow dependency.

A CLD given together with a labeling of its factors and a labeling of its links is called a *labeled CLD*, if the loop condition is satisfied. We will denote a labeled CLD by $(V, S, F; E, M; \varphi; D)$.

This definition is general, since the labels of the factors and links can be chosen independently. More special approaches will be discussed in Section 7.1.

We shall use $\longrightarrow$ and $\longrightarrow$ to denote flow and information dependencies, respectively. That is, flow dependencies are marked by a thicker arrow. Atop of the arrow we will write the link name or the link polarity when appropriate.

In case the loop condition is violated for a particular labeling of factors and links, the modeler has to choose at least one more stock along the violating loop, or apply the transformation of inserting a flow from Subsection 5.2 below.

To obtain a labeled CLD one has to decide for each factor if it is a stock, a flow, or an auxiliary. The guidance provided here is only a starting point and cannot replace a thorough introduction to system dynamics (see [Sterman, 2000], Chapter 6).

4.1. **Stocks.**

To decide whether a particular factor is a stock or not, one can ask the following questions: "Does it accumulate?" — "If time stops, can it still be measured?" — "Can I stock it somewhere and use it later?" — "Is there a level?"

All non-stocks are either flows or auxiliaries. The choice depends on the nature of the outgoing dependencies.

Once a stock s is chosen, one should look at all the incoming links of s and decide whether or not they can be labeled as flow dependency. The next two subsections discuss both alternatives.

4.2. **Flow dependencies.** Questions which characterize flow dependencies, that is links $f \longrightarrow s$ running from a flow $f \in F$ to a stock $s \in S$ include: "If s is measured with unit x, can f be measured with unit x per time unit?" — "Does the value of s at a certain time depend on the value of f at *more than one point in time* in the past?"

4.3. **Information dependencies.** Questions which characterize information dependencies $a \longrightarrow b$ for $a, b \in V$ include: "If the value of a jumps, will the value of b also jump?" — "Is a proportional to b?" — "Would the dependency still exist if I reversed the direction?" — "Can I determine the value of b at a certain time from just one value of a at the same (or earlier) time?" — "Can I reconstruct the value of a at a given time from the value of b at the same time?"

Thus information dependencies differ significantly from flow dependencies.

5. From labeled CLDs to structured CLDs

A labeled CLD can still be far from being a quantifiable SFD. However, it can be transformed in a meaningful way. The following notion describes the endpoint or goal of this transformation process.

Definition. A *structured CLD* is a labeled CLD satisfying the following properties:

(i) There are no links in S.
(ii) There are no links from A to S.
(iii) A link $e = (v_1, v_2)$ is a flow dependency if and only if $v_1 \in F$ and $v_2 \in S$, i.e. if and only if it runs from a flow to a stock.
(iv) There is no factor from F with more than two successor factors from S. In case of exactly two successors, the corresponding links must have opposite polarity[2]. In formal language, for the constellation $s_1 \xleftarrow{e_1} f \xrightarrow{e_2} s_2$, where $s_1, s_2 \in S$ and $f \in F$, we have $\varphi(e_1) = -\varphi(e_2)$.
(v) $D \cap M = \emptyset$. In other words, all links with delay must be information dependencies.

The rest of this section is devoted to the description of certain *transformations* that can be applied to turn a labeled CLD into a structured CLD.

In order to apply the transformations one traverses all links in arbitrary order. Depending on the types of start and end factor and the type of the link, the modeler has to

[2]This is the conservation of mass principle [Burns, 2001] which states that any rate node must have at least one and at most two outgoing flow links. If it has two outgoing flow links e_1 and e_2, they must have opposite polarity, i.e. $\varphi(e_1) = -\varphi(e_2)$.

This principle constraints the SFDs substantially. For example, most chemical processes can not be modeled in a straightforward way, cf. [LeFèvre, 2004]

TABLE 1. Transformations offered for different constellations on the way to a structured CLD. Depending on the labels of an arbitrary link $e = (v_1, v_2) \in E$ and the labels of its start and end factors v_1, v_2 we can distinguish the following cases. The ✓mark indicates that nothing has to be done in this case.

v_2, e	S		F		A	
v_1	M	I	M	I	M	I
S	IF,IMF	IF,CS	IS,IF,CE	✓	IS,IF,CE	✓
F	✓	IF,CE	IS,CE!	✓	IS,CE!	✓
A	IF,CA	IF	IS,CE!	✓	IS,CE!	✓

choose among the transformations given in the corresponding field of Table 1. For details on these transformations, see Sections 5.1–5.6. This choice of the transformation involves understanding of the structures to be modeled.

It is clear that after at most two iterations over all links, there will be no more transformations that can be applied.

There are three more transformations which need to be applied:

(i) Handle one-to-many flows to stocks, see Section 5.7.
(ii) Handle delays, see Section 5.8.
(iii) Beautify the CLD, see Section 5.9.

The CLD is now a structured CLD.

Of course it is not difficult to give many more transformations which simply assume that the modeler's labels contain errors and correct the labels. However, this might destroy the integrity of other links and lead away from a structured CLD. We will therefore confine ourselves to these few conversion transformations.

We will now explain the transformations in detail.

5.1. **Inserting a stock (IS).** Suppose we have the case that the CLD contains a flow dependency $e = (v_1, v_2) \in M$, or $v_1 \xrightarrow{e} v_2$, where $v_1 \in V$ is an arbitrary factor and $v_2 \in F \cup A$ is a flow or an auxiliary. This can be fixed using the following transformation, which is called *inserting a stock*.

Since a flow dependency must point to a stock but one does not want to change the type of v_2, a new stock $\tilde{s}$ (which is added to S) and a new information dependency $\tilde{e}$ (which is added to I) are introduced. The result of the transformation is

$$v_1 \xrightarrow{e} \tilde{s} \xrightarrow{\tilde{e}} v_2.$$

The new link $\tilde{e}$ has polarity $+$ in order to preserve the character of feedback loops through e and $\tilde{e}$. However, the formula of v_2 with respect to $\tilde{s}$ is not necessarily the identity; the modeler should provide it explicitly during quantification.

5.2. **Inserting a flow (IF).** In analogy to Section 5.1, we can define a transformation which will be called *inserting a flow*. Suppose that we have a link $e = (v_1, v_2) \in E$ for some $v_1, v_2 \in V$. Depending on the type of e we distinguish two cases.

5.2.1. *Inserting a flow for information dependencies.* Suppose that e is an information dependency, or equivalently, $v_1 \xrightarrow{e} v_2$, which means that e is running from v_1 to v_2. After applying the transformation this part of the diagram reads

$$v_1 \xrightarrow{e} \tilde{f} \xrightarrow{\tilde{e}} v_2,$$

where $\tilde{f}$ is a new factor which is added to F and $\tilde{e}$ is a new flow dependency.

The new link $\tilde{e}$ has polarity $+$ in order to preserve the character of feedback loops through e and $\tilde{e}$.

5.2.2. *Inserting a flow for flow dependencies.* It remains to consider the case when e is a flow dependency, i.e. $v_1 \xrightarrow{e} v_2$. This time the transformation results in

$$v_1 \xrightarrow{\tilde{e}} \tilde{f} \xrightarrow{e} v_2,$$

where $\tilde{f}$ is a new factor which is added to F and $\tilde{e}$ is a new information dependency.

The new link $\tilde{e}$ has polarity $+$ in order to preserve the character of feedback loops through $\tilde{e}$ and e. However, the formula of $\tilde{f}$ with respect to v_1 is not necessarily the identity; the modeler should provide it explicitly during quantification.

5.3. **Introduce material flow (IMF).** Consider the case of a flow dependency $e = (s_1, s_2) \in M$ running between stocks $s_1, s_2 \in S$, i.e. $s_1 \xrightarrow{e} s_2$. The intended meaning might be the that some material is taken from s_1 and put into s_2. In this case, we need to introduce a new flow $\tilde{f}$ that drives s_1, s_2 by two new flow dependencies of opposite polarity. The link e will be deleted from the diagram. The resulting part of the diagram is

$$s_1 \xleftarrow{-} \tilde{f} \xrightarrow{+} s_2.$$

The choice of the polarities describes the direction of the material flow. The problem here is that the new flow $\tilde{f}$ will not have any incoming links right after transformation described here. The modeler must connect it her/himself.

Finally, the delays have to be taken into account. If the deleted link e had been a delay link, we need to insert *two* new flow factors which are connected by an information link which is added to the set D.

5.4. **Conversion of an auxiliary to a flow (CA).** The type of any auxiliary can be changed to flow without going back on the way from a labeled CLD to a structured CLD. However, such a conversion of type cannot be done automatically, since it might add semantic errors to the model.

5.5. **Conversion of a stock (CS).** This concerns the case of an information dependency $e = (s_1, s_2) \in I$ between two stocks $s_1, s_2 \in S$. Both stocks may represent two measures of the same real world value. Relabel one stock (say s_1) as auxiliary and replace with an information dependency:

$$s_1 \longleftarrow s_2$$

This relabeling may entail new syntactic errors, which have to be fixed in the second iteration through all links.

It might also be possible to identify both factors and replace them with a single factor.

5.6. **Conversion of link type (CE).** Simply switch to the other link type to do this. Information dependencies will change to flow dependencies, and flow dependencies to information ones. The latter may cause problems, because the loop condition may be violated afterwards. We indicate this case by a ! sign.

5.7. **Handling of one-to-many flows to stocks.** This describes the case where a flow influences "too many" stocks via outgoing flow dependencies. An easy way to resolve this is to repeatedly apply the Transformation IF, cf. Subsection 5.2. However, some of these flows may be unwanted. In this case, the modeler might prefer the following transformation, which is called *isolation of a material flow*.

Suppose that a flow $f \in F$ is connected to some stocks $s_i \in S$, where $1 \leq i \leq n$ and $n \geq 2$. Choose two flow dependencies which describe a material flow from s_j to s_k for some $1 \leq j, k \leq n$. Then introduce a new flow f_{jk} and connect the original flow dependencies to this new flow. Finally, by connecting f to f_{jk} we obtain

$$\begin{array}{c} f \\ \downarrow \\ s_j \overset{-}{\longleftarrow} f_{jk} \overset{+}{\longrightarrow} s_k. \end{array}$$

When no more pairs of flow dependencies can be identified, one can apply the Transformation IF for the remaining flow dependencies.

The delays are treated in the following way: For each of the original flow dependency with delay, the delay is transferred to the newly inserted information dependency.

5.8. **Handling of delays.** The CLD may contain delayed dependencies, these must be transformed to match the SFD structure. The transformation depends on the label of the delayed dependency: Information dependencies do not need any changes. However, things change in case of a flow dependency, since material flows in SFDs can not have delays.

Suppose we have such $e = (f, s) \in D \cap M$ running from a flow $f \in F$ to a stock $s \in S$. We assume that Transformation 5.7 has already been applied to f. Then f has at most two outgoing flow dependencies, one of which is e.

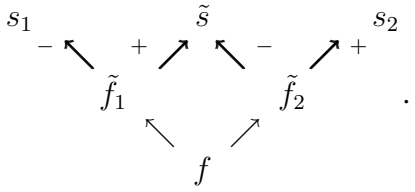
If e is the only outgoing flow dependency of f, then the constellation $f \xrightarrow{e} s$ is changed to

$$f \xrightarrow{\tilde{e}} \tilde{f} \xrightarrow{e} s$$

by applying the Transformation IF. The delay is transferred from e to the new information dependency $\tilde{e}$, meaning that $\tilde{e}$ enters the set D with the same delay value as e, and e is removed from D.

Things are slightly more complicated when f drives two stocks s_1 and s_2, since one might also need the material in transit as a factor. In this case

is transformed to



.

Here $\tilde{f}_1$, $\tilde{f}_2$ are new flow factors and the new stock $\tilde{s}$ gives the amount of material in transit. Also, two new flow dependencies with opposite polarities and two information dependencies $\tilde{e}_1$, $\tilde{e}_2$ are added to the CLD. Again, the delay from the original flow dependency e_i is transferred to the corresponding new information dependency $\tilde{e}_i$ for $i = 1, 2$. In case the material in transit is not needed, one can delete the factor $\tilde{s}$ and its adjacent links.

5.9. **Beautification.** When there are no more transformations to be applied on the CLD, the following changes can be made automatically.

(i) The label of each stock with no incoming links should be changed to auxiliary.
(ii) Each flow with no outgoing flow dependencies should be changed to auxiliary.

6. Equivalence of SFDs and structured CLDs

A structured CLD contains enough information to convert it into an SFD. In this section we will show that they are the same and describe how one can construct one from the other.

The meaning of stocks, auxiliaries and information dependencies is exactly the same in both diagrams. It remains to relate flow dependencies and material flows. To do this, we look at the cause-and-effect structure of a material flow $\boxed{s_1} \overset{f}{\Longrightarrow} \boxed{s_2}$ between two stocks $s_1, s_1 \in S$. Obviously, s_1 is not the cause for s_2. Instead, the flow f causes s_1 to decrease and s_2 to increase. The corresponding cause-and-effect structure is therefore

$$s_1 \xleftarrow{-} f \xrightarrow{+} s_2.$$

This correspondence is fundamental for understanding the relationship between CLDs and SFDs, see also [Sterman, 2000], Section 6.1.3. The drawbacks of CLDs are discussed in detail in [Richardson, 1986] and [Richardson, 1997].

We use this correspondence to convert structured CLDs into SFDs and vice versa. Any SFD can be converted into a CLD with the following transformation:

$$\boxed{s_1} \overset{f}{\Longrightarrow} \boxed{s_2} \quad \text{is equivalent to} \quad s_1 \overset{-}{\longleftarrow} f \overset{+}{\longrightarrow} s_2,$$

$$\text{☁} \overset{f}{\Longrightarrow} \boxed{s} \quad \text{is equivalent to} \quad f \overset{+}{\longrightarrow} s,$$

$$\boxed{s} \overset{f}{\Longrightarrow} \text{☁} \quad \text{is equivalent to} \quad s \overset{-}{\longleftarrow} f.$$

Formally, a structured CLD $(V, S, F; E, M; \varphi; D)$ defines an SFD $(V, \tilde{S}, F; m, I; D)$ in the following way:

$$\begin{aligned}
\tilde{S} :=& S \cup \{\text{☁}\},\\
m(f) =& \begin{cases} (s_1, s_2); & \text{if} \quad \exists (f, s_1), (f, s_2) \in E : \varphi(f, s_1) = -1 \text{ and } \varphi(f, s_2) = 1,\\ (\text{☁}, s); & \text{otherwise if} \quad \exists (f, s) \in E : \varphi(f, s) = 1,\\ (s, \text{☁}); & \text{otherwise if} \quad \exists (f, s) \in E : \varphi(f, s) = -1, \end{cases}\\
I :=& E \setminus (F \times S).
\end{aligned}$$

Conversely,

$$\begin{aligned}
S :=& \tilde{S} \setminus \{\text{☁}\},\\
M :=& \{(f, s) \in F \times S \,|\, \exists s_1 \in S : m(f) = (s, s_1) \text{ or } m(f) = (s_1, s)\},\\
E :=& I \cup M,\\
\varphi(f, s) :=& \begin{cases} -1; & \exists s_1 \in S : m(f) = (s, s_1),\\ 1; & \exists s_1 \in S : m(f) = (s_1, s),\\ \text{undefined}; & \text{otherwise.} \end{cases}
\end{aligned}$$

The polarity map is only defined for flow dependencies of the CLD. When constructing a structured CLD from a *quantified* SFD, it would be possible to examine the partial derivatives to find the polarities for information dependencies. However, such a φ depends on the state of the dynamical system. This topic has been studied in [Richardson, 1995].

7. Discussion

7.1. **Special approaches for labeling CLDs.** The definition of a labeled CLD is general; it permits to choose the labels of factors and links independently. It might be desirable to reduce this freedom, thus providing more guidance to the user and reducing the complexity of the software tool. We describe easier ways to define a labeled CLD here. To "perform labeling" there are the following actions:

(i) Prescribe the set of stocks S,
(ii) prescribe the set of flows F, or

(iii) prescribe the set of information (resp. flow) dependencies.

Any selection of two of the three actions suffices to define a labeling of a CLD (provided the loop condition is satisfied). We will discuss the three possible selections and other ideas.

7.1.1. *Label factors only.* In this case the modeler ignores all the links, she/he only labels the factors. The links are labeled by

$$M := \{(v_1, v_2) \in E \,|\, v_1 \in F \text{ and } v_2 \in S\}, \qquad I := E \setminus M.$$

This also reduces Table 1, see below.

TABLE 2. When labeling factors only, Table 1 simplifies as follows. ($\times$ denotes that this case cannot occur.)

v_2, e	S		F		A	
v_1	M	I	M	I	M	I
S	$\times$	IF,CS,IMF	$\times$	✓	$\times$	✓
F	✓	$\times$	$\times$	✓	$\times$	✓
A	$\times$	IF	$\times$	✓	$\times$	✓

7.1.2. *Label stocks and links.* Alternatively, one could select (i) and (iii), thus choosing stocks and the types of all links. We can define the missing set of flows F by taking the non-stock starting factors of all flow dependencies. Formally, this reads

$$F := \{v \in V \setminus S \,|\, \exists\, w \in V : (v, w) \in M\}.$$

The modeler no longer needs to know about the distinction between auxiliaries and flows, see Table 3.

TABLE 3. When labeling stocks and links, Table 1 simplifies as follows.

v_2, e	S		F, A	
v_1	M	I	M	I
S	IF,IMF	IF,CS	IS,IF,CE	✓
F	✓	IF,CE	IS,CE!	✓
A	$\times$	IF	$\times$	✓

7.1.3. *Label links only.* We can define the missing set of stocks S by taking the end factors of all flow dependencies. In formal language,

$$S := \{v \in V \,|\, \exists\, w \in V : (w, v) \in M\}.$$

The flows are then defined as in subsection 7.1.2.

TABLE 4. When labeling links only, Table 1 simplifies as follows.

v_2, e	S		F, A	
v_1	M	I	M	I
S	IF,IMF	IF,CS	×	✓
F	✓	IF,CE	×	✓
A	×	IF	×	✓

7.1.4. *Label stocks only.* In this simple case, the modeler only has to choose all stocks in the CLD. To obtain a labeled CLD, the choice

$$F := \{v \in V \setminus S \,|\, \exists\, s \in S \text{ such that } (v, s) \in E\}$$

of the flows is made automatically. Then, the links are labeled automatically as described in Section 7.1.1. A labeled CLD generated this way will already be close to a structured CLD; the problem remaining is that of two stocks connected by a link. Here one of the transformations IF, IMF or CS must be considered.

7.2. **Software implementation.** We are currently implementing a software application which aims at supporting the whole modeling workflow. This workflow is based on experience with teaching System Dynamics at Lund University. It comprises the following worksteps, where each workstep has a well-defined goal:

Define asks for a problem statement, time horizon and aims. This serves as an anchor and helps the modeler keep focus in the following steps.
Explore identifies the key factors of a system.
Connect builds a CLD from these key factors (and possibly others)
Quantify transforms the CLD into a structured CLD or SFD and requests equations and/or data for all factors.
Simulate uses the quantified model to create charts which show the behavior over time and lets the user compare simulation results with reference behaviors.

These steps do not have to be followed in a strict sequence, since we believe that this would constrict creativity. This allows to work incrementally, that is to start with a small number of factors, build a small working model and then add more information iteratively.

Each workstep is connected with a *perspective* which shows only the necessary tools and data for the task at hand to the modeler.

Incomplete or inconsistent data does not prevent the modeler from proceeding or changing the workstep. Instead, the software analyzes the project data and maintains a list of *issues*, similar to a todo-list. Each issue describes what is wrong and how to correct it. If possible, it also offers *quick fix actions* to the modeler which automate the correction.

7.2.1. *Transition from a CLD to an SFD.* The four phases described in Section 1 are supported by our software in the following way:

(i) The CLD is created in the workstep *Connect*. After switching to the workstep *Quantify*, the modeler is asked to label all factors as stocks or non-stocks.
(ii) At the same time, the list of issues is updated with all violations of the constraints for a structured CLD and the offending elements are marked in the diagram. The modeler corrects these issues one by one, possibly with the help of quick fix actions. At any time the list of issues provides a systematic guidance and gives feedback about the diagram's syntactic quality.
(iii) If no constraints are violated, the modeler may switch from the structured CLD view to the SFD view. The structured CLD ist transformed automatically into the corresponding SFD.The layout of the CLD is kept as far as is possible.
(iv) Quantification may happen either in the SFD view or the structured CLD view. Missing formulas, missing input values, unit inconsistencies and any other quantification errors are reported as issues. Again, quick fix actions are offered if possible (for example, using the simplest formula which satisfies unit consistency). Therefore, the modeler receives constant feedback about what has to be done before the model may be simulated.

The modeler is not forced to proceed sequentially through these phases. At each point in the process, the she/he is free to make arbitrary changes to the model and restart the process at an earlier phase. Obviously by doing so the number of issues might increase again.

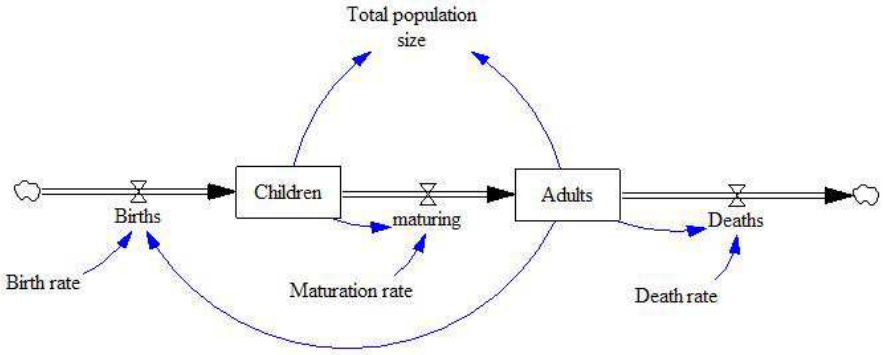
We think that this procedure allows a smooth transition from CLDs to SFDs. It is not only possible to reuse most of the CLD for the SFD, but the software also gives feedback about the progress of the transition. However, this progress can only be measured by the syntactic correctness of the model. Validation, that is comparison with the real world system to be modeled, happens at a later phase in our workflow.

8. Another Example: Population

We use the model of a simple child-adult population model as an example to find out if different initial CLDs can be transformed into the same resulting SFD. For this we start by looking at the result in Figure 7 and then work with some CLDs which might have been the starting point for this model.

The model consists of two stocks, *Children* and *Adults*, three flows which represent the births, the children reaching maturity, and the deaths. Finally, there are some auxiliaries, which serve as input and output factors. A structured CLD which is equivalent to the SFD is shown in Figure 8. This transition is automatic apart from the fact that it is not possible to obtain polarities for information dependencies when going from the SFD to the structured CLD. The labels of the factors are supposed to be the same as in the CLD; the labels of the links are displayed in the diagram.

FIGURE 7. The SFD of a child-adult population model.



For a simple process like children being born, maturing, and growing old, the CLD in Figure 8 is already too complicated to be obvious. A modeler will usually start with a simpler CLD, and develop it further. Figure 9 (a) shows an extreme example of such a CLD. There are just the two stocks with two dependencies. After applying the transformation IMF from Section 5.3 and IF from Section 5.2 to the flow and information dependency, respectively, the CLD is a structured CLD. However, it is still incomplete semantically; the modeler discovers the need of auxiliary factors and information dependencies and inserts them. Presumably, she/he would do this correctly, in which case there is no deviation from a structured CLD, hence no further interaction about transformations would be necessary.

FIGURE 8. The structured CLD which is equivalent to the SFD in Figure 7.

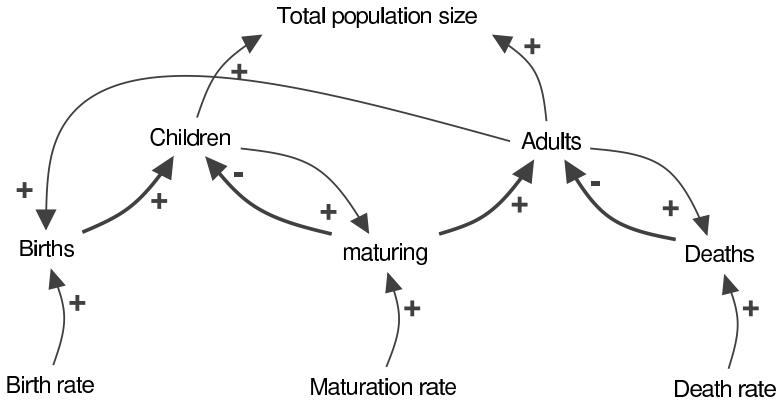
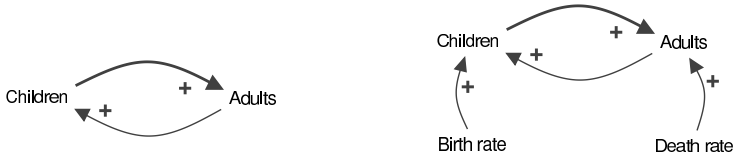


FIGURE 9. (a) A possible (if extreme) example; (b) a problematic case.



(a) (b)

Figure 9 (b) shows an example which does not lead to the desired structured CLD. After applying the corresponding transformations from example (a) the information dependencies starting at *Birth rate* and *Death rate* will still point to stocks. Table 1 suggests the IF transformation for these links again. At this point, the modeler would have to think again about these dependencies. She/he would realize that the rates influence the flows instead of the stocks and change the end factors of the two information dependencies.

This reveals an aspect giving opportunity for refinement of the algorithm: The transformations adding factors to the model, as IMF, IS, and IF, could also take into the account other links running into or out of the factors adjacent to the link that is considered.

The example shows that there is some similarity to the component approach formalized in [Burns & Ulgen, 2002a] and [Burns & Ulgen, 2002b]. One could consider the children and adults stocks as components which are expanded during the development of the model.

9. SUMMARY

In this paper we described a method for transforming CLDs to SFDs systematically. We provided guided transformation steps which help in this process.

CLDs are an accessible tool for modeling complex systems. They can be more compact than a corresponding SFD since it is not necessary to include every flow. Also, they contain a minimum of formal notation symbols, there are no double lines and system boundary symbols which "clutter" the diagram. CLDs are the appropriate view for analyzing feedback loop structures. However, the price for this abstraction is the lack of formal exactness, which hinders the direct interpretation as a quantitative model.

Structured CLDs have this exactness but also include additional assumptions. These assumptions are hard to check by just looking at the diagram, and may take the same efforts as performing this step manually without guidance. We developed a tool which does these checks automatically and which can simulate a structured CLD as a model. In fact, a structured CLD is the same as an SFD up to notational differences, which

allows automatic conversion between a structured CLD and an SFD. One of the major advantages lies in the possibilities to guide the modeler through this crucial step.

Work is underway to implement the described interactive algorithm in a software tool. This allows to use accessible CLDs for exploration and for discussions with non-experts. The distinction between stocks and rates takes place during a second stage, when a rough structure of the modeled system is already available. The transition from a CLD to a structured CLD can already be combined with validation and quantification.

In conclusion the suggested method may be a good method to implement in a system dynamics software solution for overcoming one of the obstacles in the system dynamics modeling route. It will guide the modeler through this phase and thus improve the modeling process.

REFERENCES

[Burns, 2001] Burns, J. R. (2001). Simplified translation of CLDs into SFDs. In *Proceedings of the 19. International Conference of the System Dynamics Society, Atlanta, GA, July 2001*.

[Burns & Ulgen, 2002a] Burns, J. R. & Ulgen, O. (2002a). A component strategy for the formulation of system dynamics models. In *Proceedings of the 20. International Conference of the System Dynamics Society, Palermo, Italy, July 2002*.

[Burns & Ulgen, 2002b] Burns, J. R. & Ulgen, O. (2002b). A matrix architecture for development of system dynamics models. In *Proceedings of the 20. International Conference of the System Dynamics Society, Palermo, Italy, July 2002*.

[Ford, 1999] Ford, A. (1999). Modeling the environment: An Introduction to System Dynamics Models of Environmental Systems. Island Press, Washington, DC.

[Haraldsson, 2004] Haraldsson, H. V. (2004). Introduction to systems thinking and causal loop diagrams. Technical report, Lund University, Department of Chemical Engineering. Reports in Ecology and Environmental Engineering, pp. 1–49.

[Haraldsson & Sverdrup, 2003] Haraldsson, H. V. & Sverdrup, H. (2003). Finding simplicity in complexity in biogeochemical modelling. In Wainwright, J. & Mulligan, M., (eds.) *Environmental Modelling: Finding Simplicity in Complexity*. Wiley, New York, pp. 211–213.

[Homer & Oliva, 2001] Homer, J. & Oliva, R. (2001). Maps and models in system dynamics: a response to Coyle. *System Dynamics Review*, **17**, 347–355.

[LeFèvre, 2004] LeFèvre, J. (2004). Why and how should we replace the tank-pipe analogy of our stock flow models by a chemical process metaphor. In *Proceedings of the 22. International Conference of the System Dynamics Society, Oxford, United Kingdom, July 2004*.

[Randers, 1980] Randers, J. (1980). Elements of the System Dynamics Method. The MIT Press, Cambridge, MA.

[Richardson, 1986] Richardson, G. P. (1986). Problems with Causal Loop diagrams. *System Dynamics Review*, **2**, 158–170.

[Richardson, 1995] Richardson, G. P. (1995). Loop polarity, loop dominance, and the concept of dominant polarity. *System Dynamics Review*, **11**, 67–88.

[Richardson, 1997] Richardson, G. P. (1997). Problems in Causal Loop diagrams revisited. *System Dynamics Review*, **13**, 247–252.

[Richardson, 1999] Richardson, G. P. (1999). Reflections for the future of system dynamics. *J. of the Operational Research Society*, **50**, 440–449.

[Sterman, 2000] Sterman, J. D. (2000). Business Dynamics Systems: Thinking and modeling for a complex world. McGraw-Hill.

[Wolstenholme, 1999] Wolstenholme, E. F. (1999). Qualitative vs. quantitative modelling: the evolving balance. *J. of the Operational Research Society*, **50**, 422–428.

INSTITUTE FOR NEURO- AND BIOINFORMATICS, UNIVERSITY OF LÜBECK, RATZEBURGER ALLEE 160, D-23538 LÜBECK, GERMANY

E-mail address: binder@inb.uni-luebeck.de

INSTITUTE FOR SOFTWARE TECHNOLOGY AND PROGRAMMING LANGUAGES, UNIVERSITY OF LÜBECK, RATZEBURGER ALLEE 160, D-23538 LÜBECK, GERMANY

E-mail address: vox@isp.uni-luebeck.de

DEPARTMENT OF CHEMICAL ENGINEERING, LUND UNIVERSITY, P. O. BOX 124, S-221 00 LUND, SWEDEN

E-mail address: salim.belyazid@chemeng.lth.se

DEPARTMENT OF CHEMICAL ENGINEERING, LUND UNIVERSITY, P. O. BOX 124, S-221 00 LUND, SWEDEN

E-mail address: hordur.haraldsson@chemeng.lth.se

CENTRE FOR ENVIRONMENTAL STUDIES, MICLU, LUND UNIVERSITY, P. O. BOX 170, S-221 00 LUND, SWEDEN

E-mail address: mats.svensson@chemeng.lth.se