Modeling of Resilient Systems in Default Logic

Andrei Doncescu[1]
[1] LAAS-CNRS/University of Toulouse
Toulouse France

andrei.doncescu@laas.fr

Abstract. In this paper we propose a reconfiguration model based on switched flat system. The interest to have flat subsystems is to assure the property of transitivity. Transitivity is one the key points of a resilient system keeping the dependability. To reconfigure the system in the case of unexpected phenomena we use default logic.

1 Introduction

In this paper we present a reconfigurable model of resilient systems. The reconfiguration is an important method of resilient systems keeping stability. This approach could be applied to a large category of systems having a nonlinear dynamic, from biological systems to robots and aircraft. What characterizes all these systems is the high complexity. The increasing complexity makes systems more and more vulnerable for faults and chaotic behavior. The system state may either evolve continuously for some duration of time according to one set of differential equations or be abruptly reset to a new value from which evolution is governed by another set of differential equations. The commutations are typically triggered by the occurrence of some discrete event.
During the last decades the adjective Resilient has been used for labeling the systems, which are faults tolerant but ignoring the unexpected aspect of the phenomena that the systems have to face, therefore the necessity of a fault-diagnosis and fault-tolerant control. Monitoring and diagnosis of any resilient system depend on the ability to estimate the system state given the observations. Estimation for hybrid systems is particularly challenging because it requires keeping track of multiple models and the transitions between them.
The different approaches are related to the a priori representation of the knowledge. The physical models basically represented by differential equations “mime” physical structure and give a synoptic view. The engineering aspect is defined by functional models, which describe the chain of functions realized by the system. The representation of knowledge about the system leads to other type of models: informational, which are supposed to gather signals and find out the relations causality/effects.
Our viewpoint is all complex or resilient systems could be modeled by hybrid dynamical subsystems. Therefore the state may either evolve continuously for some

duration of time according to one set of differential equations or be abruptly reset to a new value from which evolution is governed by another set of differential equations, with the switches typically triggered by the occurrence of some discrete event, therefore the signal abstraction could be very useful. Two types of data exist in generic databases describing the hybrid systems: numerical and symbolic.
In the case of hybrid dynamic systems the quantity of data describing the evolution of the complex system can be very important and difficult to figure out the analytical model therefore a supervised learning model seems to be the only solution.
We point out the problem of discretization, which influences the results either by an over fitting (i.e. finding meaningless regularity in data due to a large number of possible hypotheses) or by missing important events.
The resilience is the property of a complex system to successfully recover environmental perturbations or disturbances. Contrary, of the feeling that stability is a property of resilient systems, resilience is one of the properties of stable dynamic systems.
The misunderstandings and problems that continue to occur will eventually cause fatal damage to the system must be avoid by the construction or modeling of resilient systems.
The notion of resilience has been introduced in different fields:

1. in ecology [4], referring to moving from a stability domain to another one under the influence of disturbances;
2. in business [5], referring to the capacity to reinvent a business model before circumstances force to;
3. in industrial safety [6], referring to anticipating risk changes before damage occurrence.

Our definition of resilience is:
"The capacity of a complex system to react in presence of disturbances by switching from one dynamical model to another one by keeping the global stability properties".

The main idea of flatness is to connect the different subsystems in a new configuration.

2. Flat Systems

There is a three-step process for describing equations of physics that is often helpful in clarifying the distinction between different types of ideas. The first step is to describe the kinematics of the process, i.e. the basic variables in the problem and the physically inherent restrictions of them. Next, one poses universal laws that govern all processes of the type under consideration. Finally, one postulates constitutive laws that differentiate one physical situation from another.
In the case of resilient systems we should be able to determine the state of the system and to control it from the outputs. A special type of systems named flat satisfies this request. Intuitively, a system is said to be differentially flat if a set of variables called

flat outputs can be found for which all states and actions can be determined from them without integration.

A general nonlinear system given by :

$$\dot{\underline{X}} = F(\underline{X},\underline{U}),\ \underline{X}\in\mathbf{R}^n,\ \underline{U}\in\mathbf{R}^m, \qquad \text{(A-1)}$$

where is a smooth mapping, is said explicitly flat with respect to the output vector , if is an order vector which can be expressed analytically as a function of the current state, the current input and its derivatives, while the state and the input vectors can be expressed analytically as a function of and a finite number of its derivatives. Then there exists smooth mappings G_X, G_U, and G_Z such as:

$$\begin{cases} \underline{Z} = G_Z\left(\underline{X},\underline{U},\ldots,\underline{U}^{(n_z)}\right) & \text{A-2} \\ \underline{X} = G_X\left(\underline{Z},\dot{\underline{Z}},\ldots,\underline{Z}^{(n_x)}\right) & \text{A-3} \\ \underline{U} = G_U\left(\underline{Z},\underline{Z},\ldots,\underline{Z}^{(n_x+1)}\right) & \text{A-4} \end{cases}$$

where n_z and n_x are integer numbers. Vector $\underline{Z}$ is called a flat output for the nonlinear system. There is no systematical way to determine flat outputs and eventually to prove its uniqueness, the flat outputs usually possess some physical meaning.

The explicit flatness property is of particular interest for the solution of control problems when physically meaningful flat outputs can be related with their objectives. In many situations, the control problem can be formulated as a flat output trajectory following problem. In general, for these cases, the flat output of equation (A-2) can be reduced, through state transformation, to a function of a single argument, the new system state itself:

$$\underline{Z} = G_Z\left(\underline{X}\right) \qquad \text{A-5}$$

We would like to make the dissociation between resilience and stability: it is noted that “a system can be very resilient and still fluctuate greatly, i.e., have high stability” and that “high stability seems to introduce high resilience”;

2 Modeling of Switched Systems

We have considered in this models that “Switched systems are more than the sum of their subsystems”, which is the most important property of complex and resilient systems. A switched systems is represented:

$V = U \cup Y \cup X$: Input, output, and internal (state) variables

Q: States, a set of valuations of X

$\Theta \subseteq Q$: Start states

$A = I \cup O \cup H$: Input, output, and internal actions

$D \subseteq Q \times A \times Q$: Discrete transitions

T: Trajectories for V.

3 Causality and Classical Inference

If the inference of classical logic A → B or A ⊢ B is fully described formally, with all the "good" logic properties (tautology, not contradiction, transitivity, contraposition, modus ponens, ...), a description of the properties of causality is not simple. Causality cannot be seen as a classical logic relation.
A basic example is "If it rains the grass is wet". This expression cannot be translated by the formula *Rain → lawn-wet*, which means if *it rains the grass is always wet*. Indeed, there may be exceptions to this rule (the lawn is under a shed ...). You can also change the environment (we cover the lawn).
The rules with exceptions are well known in Artificial Intelligence. They drive, in particular, to nonmonotonic logics and revision theories. On the other hand and more technical, we find here all the classic problems that arise when one wants to try to formalize and use of negation by failure in programming languages such Solar [3]. In this paper we describe a very simple and efficient form of causality necessary and probably sufficient for the application to complex and resilient systems.
To describe interactions between subsystems we use a language L of classical logic (propositional or first order logic). The proposition A (resp. ¬ A) says that A is true (false).
If the system is subject to some unexpected perturbations represented as *reability → ¬perturbation*, could be interpreted by « something » protects against perturbations. We are in a logical framework, so it is possible to represent almost everything in a natural way. But the price to pay is the complexity. If you use the entire first order language can be the combinatorial explosion of algorithms and incompleteness.
The goal of this paper is the interactions between subsystems view as a very simple form of causality. To express these interactions it is common to represent by two binary relations *connect(A,B) and failed(A,B)*. The first relation means, for example, a subsystem A stands of a subsystem B. The second relation is a failure. Conventionally, these relations are represented by $A \Rightarrow B$ *and* $A \dashv B$. Of course, this causality is basic and a lot of research papers describe this type of representation of the causality.
Depending on the context, true could be interpreted as known, certain, believed ... or, more technically in a system of automated theorem proved.
The first idea is to express these laws in classical logic by axioms:

$cause(A, B) \wedge A \Rightarrow B$
$failed(A, B) \wedge A \Rightarrow \neg B$

Therefore, to provide the causal links between our relations connect and failed in a classical language (propositional calculus or first order logic) it is necessary to describe :

1. the internal characteristics of relations and cause and block failure
2. the links between these relations and classical logic

They can also be weakly expressed more by rules of inference, close to Modus Ponens :

$cause(A, B), \ A \vdash B$
$failed(A, B), \ A \vdash \neg B$

But these two formulations are problematic when a conflict appears.
For example, a set of four formulas *F = {A, B, cause(A, C), failed(B, C)},* leading to infer from F, B and ¬ B and this is inconsistent. To solve such conflicts, we can try to use some methods inspired by constraint programming, as the negation by failure.
It is also possible to use a defeasible reasoning, especially a nonmonotonic logic. The first method (negation by failure) poses many theoretical and technical problems if you leave the simple cases. These problems are often solved by adding properties to the formal system, properties that pose other problems.

3.1. Causality and default logic

To resolve conflicts seen above, the intuitive idea is to lighten the formulation of rules of causality:

(1 ') If A causes B, if A is true, and it is possible that B, then B is true.
(2 ') If A blocks B, if A is true, and it is possible that B is false then B is false.

The question then is to describe as formally as possible. This question began to arise in artificial intelligence thirty years ago, when it was formalized the natural human reasoning. In this type of reasoning, it is necessary to reason with incomplete information, uncertain and subject to revision and sometimes false information. On the other hand we have to choose between several possible conclusions contradictory. The basic example is: {The penguins are birds, birds fly, penguins do not fly}. If Tweety is a penguin we arrive at a contradiction, the system is inconsistent. This inconsistency can be ignored if we can handle the exception by replacing "Birds fly" with "Typically birds fly". The nonmonotonic logic formally describes the modes of reasoning that takes into account these phenomena.
To represent the reconfiguration of resilient systems we propose to use default logic of Reiter. In this logic, the rules (1) and (2) will be expressed intuitively.

(1) If A causes B, if A is true, and if B is not contradictory, then B is true.
(2) If A blocks B (because A failed), if A is true, and if ¬ B is not contradictory then ¬ B is true.
In default logic, these rules can be represented by normal defaults and written:

$d1 = A : B / B$

$d2 = A : \neg B / \neg B$

Therefore, the information is represented here using defaults theory $\Delta = \{W, D\}$, where *W* is a set of classical logic formula and is the set of defaults used to represent the uncertainty of some information.
The classical definition of extension is based on the utilization of *W* and a subset of defaults *D*. The condition to use a default starts by checking the prerequisites are satisfied and the consequence doesn't lead to contradiction. In a simple manner that means his negation is not verified. If this request is *TRUE* we add the consequence to *W* and the algorithm is restarted until all defaults has been used.
For example, consider $\Delta = \{W, D\}$ *with* $W = \{ A \}$ *and* $D = \{d1 , d2\}$.

The 2 extensions are :

E1 = { A, B} if d1 is used.

E2 = { A, B} if d2 is used.

By using default logic, the conflict is resolved, but it is not possible to rank the extensions: B is true or false ? In fact this will really depend on the context. Some times the positive interactions are preferred to negatives. Another possibility is to use probabilistic or statistical methods or to weight each extension based on the evaluation of the knowledge. From algorithmic viewpoint of the ranking of extension could be evaluated also during the calculation of the extensions even the off-line ranking is preferred.

4 Representation of Resilient Systems Reconfiguration

How it is described above the defaults are used to manage incomplete information. Its most general form, a default is an expression of the form:

$$D=(A_x(X):B_y(X) \wedge C(X))/(C(X)) \qquad \text{A-5}$$

where $A_x(X)$, $B_y(X)$ and $C(X)$ ($x = 1,2, ..., m,\ y = 1,2, ..., l$) are well-formed formulas which contain first order as free variable X or $X = (x_1, x_2, x_3, \ldots, x_n)$ as a vector of free variables. $A_x(X)$ are the prerequisites, $B_y(X)$ are the justifications and $C(X)$ is the consequent.

The default (A-5) means informally: if $A_x(X)$ are verified (at some moment t_i), if possible that $B_y(X)$ are real ($B_y(X)$ are consistent), and if possible that $C(X)$ is true (at the moment t_i+1), then we infer $C(X)$ (at the moment t_i+1).

The use of defaults increases the number of formulas derived from the knowledge base W: we get extensions that are sets of theorems derivable monotonically.

An extension of the default theory $\Delta = (D, W)$ is a set E of formulas, closed for the deduction, containing W and satisfying the following property: if d is a default of D whose prerequisites $A_x(X, t_i)$ are in E, without the negation of justifications $B_y(X)$ and of consequent $C(X, t_{i+1})$ are in E, then the consequent of d is in E.

Formally, the extensions are defined as follows:

E *is an extension of* Δ *iff* $E = \bigcup_{i=0,\infty} Ei$, *with*

$E_0 = W$

and for $i > 0$,

$$E_{i+1} = ThE_i \cup \{C(X,t_{j+1}) / \frac{(A_x(X):B_y \wedge C(X))}{C(X)} \in D, A_x(X) \in E_i\left(at\ t_j\right), \neg By \notin Ei,$$

$$\neg CX \notin Ei\left(at\ t_{j+1}\right)\}$$

where $Th(Ei)$ *denotes the set of theorems obtained monotonically from*

$E_i : ThE_i = \{w / E_i \vdash w\}$.

The calculation of extensions allows to study the defaults one by one and to retain those who respond to the problem and are compatible with each other. Each extension corresponds to a possible solution of the problem. To calculate an extension, we must verify that the negation of justification does not belong to E_i. We can therefore use an incremental algorithm for computing extensions.

For a default theory $\Delta = (D, W)$, with the set of defaults D and the knowledge base W, the calculation is extended according to the algorithm:

```
Input : E=θ; (set of extensions E is empty).
Output : E=U(i=0,N) Ei.
calcul_extension(E) :
{
while there is a default D=(Ax (X):By(X)∧C(X))/(C(X))
that has not yet been inspected do
    - Select the default D,
    - Verify that the prerequisites Ax(X) are true (at
    some moment tj),
    - Verify that the justifications By(X) are
    consistent with W,
    - Verify that the consequent C(X) is consistent
    with W (at the moment tj+1),
    - Add By(X) and C(X, tj+1) to W.
end while
End of the calculation for an extension.
Backtracking (Deleting the last C(X,tj+1) and By(X) added
to W).
calcul_extension(E).
}
```

In our model, to provide links between these subsystems active and non-active by failure, the intuitive idea is to weaken the formulation of 3 causation rules:

(1) *If*
system(A,ON,t_i) , connect(A,B) and connect(B,C) are true,
and if
it is possible that reliable(A,B), non_reliable(B,C) and system(B,ON,t_{i+1}),
then
system(B,ON, t_{i+1}) is true.

(2) *If*
system(A,ON,tj), connect(A,B), and connect(B,C) are true,
and if
it is possible that not_reliable(A,B), reliable(B,C) and system(B,OFF,t_{j+1}),
then
system (B,OFF, t_{j+1}) is true.

(3) *If*
system(A,OFF,t_k), connect(A,B) are true,
and if it is possible that reliable(A,B) and system (B,OFF,t_{k+1}),
then
system(B,OFF, t_{k+1}) is true.

The predicate *reliable* has the meaning of activity of two entities and the first entity trigs the second one.
Formally the possible connectivity between 3 subsystems A,B,C are described in default logic by :

(1') *If*
system(A,ON,t_i), connect(A,B) and connect(B,C) are true,
and if
connect(A,B),non_connect(B,C) and system(B,ON,ti+1) are not contradictory,
then
system(B,ON, t_{i+1}) is true

(2') *If*
system(A,ON,tj), connect(A,B) and connect(B,C) are true,
and if
non_connect(A,B), reliable(B,C) and system (B,OFF,t_{j+1}) are not contradictory,
then
system (B,OFF, t_{j+1}) is true

(3') *If*
system(A,OFF,tk) and connect(A,B) are true,
and if
reliable(A,B) and system(B,OFF,t_{k+1}) are not contradictory,
then
system(B,OFF, t_{k+}1) is true

In default logic, these rules will be represented by the set of defaults *D* and written as:

$$d1:(system(A,ON)\wedge connect(A,B)\wedge connect(B,C):reliable(A,B)\wedge non\_reliab(B,C)\wedge system(B,ON))/(system(B,ON))$$
$$d2:(system(A,up)\wedge connect(A,B)\wedge connect(B,C):non\_reliable(A,B)\wedge reliable(B,C)\wedge system(B,OFF))/(system(B,OFF))$$
$$d3:(system(A,OFF)\wedge connect(A,B):reliable(A,B)\wedge system(B,OFF))/(system(B,OFF))$$

Therefore, the conflict has been resolved.

If we consider a plant with 5 entities *A,B,C,D,E* connected between them, and A is submitted to a perturbation. We want to know what is the possible reconfigurations of B,D,C and E.
Using default theory $\Delta = (D, W)$, in that W = {*perturbation*(A, up, t_0)}, by applying the algorithm above, we have 12 exceptions.

The following is one of them:

joint(system(A,ON,t_0),non_reliable(A,B),reliable(B,D)) -> system(B,OFF,t_1)
joint(system(B,OFF,t_1),reliable(B,C)) -> system(C,OFF,t_2)
joint(system(B,OFF,t_1),reliable(B,D)) -> system(D,OFF,t_2)
joint(system(D,OFF,t_2),reliable(D,E)) -> system(E,OFF,t_3)

This result us the worst one because the configuration of the complex systems is not able assure a healthy behavior in the case of a Fault on the subsystem A even if A keeps nominal parameters and it is considered ON.

5 Conclusion

We have introduced a new-switched system model based on a hybrid approach. To switch from one dynamic to another one we use Default Logic. The most important property, which assumes the reliability, is the flatness of the subsystems.
All these representations consider the problems of uncertain and revision. For the first aspect a minimum and necessary link between two causal relationships, it was necessary to formalize by using default logic.
All this approach offers a model of simulation for resilient systems and the future work will consider the structure network as fundamental of complex systems.

References

1. H. Goto, Y. Hasegawa, and M. Tanaka, “Efficient Scheduling Focusing on the Duality of MPL Representatives,” Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670.
2. H. Nabeshima, K. Iwanuma and K. Inoue. SOLAR: a consequence finding system for advanced reasoning. Proc. Eleventh Int. Conf. Automated Reasoning with An- alytic Tableaux and Related Methods, Proc. TABLEAUX 2003, LNAI 2796, pages 257–263, Springer, 2003.
3. C.S. Holling, "Resilence and stability of ecological systems", Annual Review of Ecology and Systematics, vol. 4, 1973, pp. 1-23.
4. G. Hamel, L. Välikangas, "The quest for resilience", Harvard Business Review, Sept. 2003.
5. E. Hollnagel, D. Woods, N. Leveson (Eds.), Resilience Engineering – Concepts and Precepts, Ashgate, 2006.