1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07)
ENS Cachan, France - June 13-15, 2007

OPTIMAL DISCRETE CONTROLLER SYNTHESIS FOR MODELING FAULT-TOLERANT DISTRIBUTED SYSTEMS

E. Dumitrescu [1] **A. Girault** [2] **H. Marchand** [3] **E.Rutten** [4]

Abstract: We propose a safe design method for safe execution systems, based on fault-tolerance techniques: it uses optimal discrete controller synthesis (DCS) to generate a correct-by-construction fault-tolerant system. The properties enforced concern consistent execution, functionality fulfillment (whatever the faults, under some failure hypothesis), and several optimizations (of the tasks' execution time). We propose an algorithm for optimal DCS on bounded paths. We propose model patterns for a set of periodic tasks with checkpoints, a set of distributed, heterogeneous and fail-silent processors, and an environment model that expresses potential fault patterns. The implementation is illustrated using the Sigali symbolic DCS tool and the Mode Automata programming language. *Copyright ©2007 IFAC*

Keywords: Optimal discrete controller synthesis, fault-tolerance, checkpointing, reactive systems, embedded systems.

1. MOTIVATION

Reactive systems must respond continuously to their environment, at a speed imposed by the latter. This property imposes a real-time constraint: the environment sends stimuli to the system, to which it must react before the next stimuli. In safety critical systems, this constraint is *strict*, and hence it must be guaranteed on the implementation. We wish to design multi-task multi-processor reactive systems. As processors are subject to failures, we make sure that our systems are fault-tolerant (safety critical constraint). We also guarantee that they react within some fixed temporal bound (real-time constraint). Our proposal is to use discrete controller synthesis (DCS) to obtain automatically a controlled system that guarantees its fault-tolerant and real-time constraints by construction.

We represent a system by its behavior in terms of starting and reconfiguring lower-level computations [Altisen et al. (2003)]. Each computation is represented as a *periodic task*, whose period matches the reaction time imposed by the environment. Each task is itself decomposed into several successive *phases*, separated by a checkpoint. A *reconfiguration* of the system consists in migrating one or several tasks onto another processor. This must be done, in particular, each time a processor failure occurs. When doing so, the task rolls-back to its last checkpoint.

The major advantage of using DCS for fault-tolerance is that, if DCS succeeds, then the resulting controlled system will *dynamically* reconfigure itself upon the occurrence of a failure, while being guaranteed *by construction* that it satisfies its real-time constraints whatever the failures.

This paper builds up upon previous results where DCS was applied with objectives of invariance, cost bounding, and *one-step* optimal DCS [Girault and Rutten (2004)]. Here, we have a *richer* task model with phases and checkpoints, and we apply optimal DCS on *finite paths*. To achieve this, we propose a variant of the classical optimal DCS algorithm of Bellman [Bellman (1957)] in order to cope with path having waiting loops, as it occurs with reactive systems.

In the remainder, we first describe our model of distributed systems with faults, then we describe our DCS technique for fault-tolerance, and particularly optimal DCS on paths dealing with self-loops encountered in reactive systems. Next we define fault tolerance in terms of properties on the model, which are used as objectives for DCS, and lead to the automatic derivation of a controller for fault tolerance.

[1] INSA-Lyon, http://www.insa-lyon.fr, emil.dumitrescu@insa-lyon.fr
[2] INRIA, LIG, http://pop-art.inrialpes.fr/~girault, Alain.Girault@inria.fr
[3] INRIA, IRISA, http://www.irisa.fr/prive/hmarchan, Herve.Marchand@inria.fr
[4] INRIA, LIG, http://pop-art.inrialpes.fr/~rutten, Eric.Rutten@inria.fr

2. DCS ALGORITHMS AND MODELS

We adopt an existing DCS framework [Marchand et al. (2000)] and propose a modeling methodology. Thus we only introduce the useful definitions or technical aspects of the tools, and summarize the functionality.

2.1 Labeled transition systems (LTS)

Formally, an LTS is a tuple $\langle \mathcal{Q}, q_0, \mathcal{I}, \mathcal{O}, \delta \rangle$, where $\mathcal{Q}$ is a finite set of states, q_0 is the initial state, $\mathcal{I}$ is a finite set of input signals (produced by the environment), $\mathcal{O}$ is a finite set of output signals (issued to the environment), and δ is the transition function, i.e., a mapping from $\mathcal{Q} \times \mathcal{B}ool(\mathcal{I}) \times \mathcal{O}^* \rightarrow \times\mathcal{Q}$. Each transition has a label of the form g/a, where $g \in \mathcal{B}ool(\mathcal{I})$ must be true for the transition to be taken (g is the guard of the transition), while $a \in \mathcal{O}^*$ is a conjunction of outputs that are issued when the transition is taken (a is the action of the transition). A transition (s, g, a, s') will be graphically noted $s \overset{g/a}{\rightarrow} s'$. We use this level of definition for our modelling work, in a graphical form in the Figures of this paper. A *path* is a sequence (possibly infinite) of transitions starting from the initial state q_0. In this paper, we only focus on LTSs which are *deterministic* and *reactive*:

- determinism guarantees that the system always reacts in the same manner to the same sequence of input events;
- reactivity guarantees sensitivity to any event feed from its environment.

Two LTSs $\langle \mathcal{Q}_1, q_{01}, \mathcal{I}_1, \mathcal{O}_1, \delta_1 \rangle$ and $\langle \mathcal{Q}_2, q_{02}, \mathcal{I}_2, \mathcal{O}_2, \delta_2 \rangle$ are said to be *compatible* only if their output sets are disjoint $\mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset$. *The synchronous product* between two compatible LTSs $\langle \mathcal{Q}_1, q_{01}, \mathcal{I}_1, \mathcal{O}_1, \delta_1 \rangle$ and $\langle \mathcal{Q}_2, q_{02}, \mathcal{I}_2, \mathcal{O}_2, \delta_2 \rangle$ is the LTS $\langle \mathcal{Q}_1 \times \mathcal{Q}_2, (q_{01}, q_{02}), \mathcal{I}_1 \cup \mathcal{I}_2, \mathcal{O}_1 \cup \mathcal{O}_2, \delta_1 \times \delta_2 \rangle$.

2.2 Discrete controller synthesis (DCS)

DCS emerged in the 80's [Ramadge and Wonham (1987)], with foundations in language theory. Its purpose is, given two languages $\mathcal{P}$ and $\mathcal{D}$, to obtain a third language $\mathcal{C}$ such that $\mathcal{P} \cap \mathcal{C} \subseteq \mathcal{D}$. This is a kind of inversion problem, since one wants to find $\mathcal{C}$ from $\mathcal{D}$ and $\mathcal{P}$. Here, $\mathcal{P}$ is called the *plant*, $\mathcal{D}$ the *desired system* or *objective*, and $\mathcal{C}$ the *controller*. Several teams proposed extensions and applications of this language theory technique to labeled transition systems (LTS).

In our approach, $\mathcal{P}$ is specified as a LTS, and $\mathcal{D}$ is an objective to be satisfied by the controlled system, typically *making a subset of states invariant* in the controlled system, or *keeping it always reachable*. The controller $\mathcal{C}$ obtained with DCS is a constraint restricting the transitions of $\mathcal{P}$, i.e., inhibiting those that would jeopardize the objective. The key point is that the set of inputs $\mathcal{I}$ is partitioned into two subsets, $\mathcal{I}_c$ and $\mathcal{I}_u$, respectively the set of *controllable* and *uncontrollable* inputs. The controller $\mathcal{C}$ can only inhibit those transitions of $\mathcal{P}$ for which the guard contains at least one controllable signal, i.e., in $\mathcal{I}_c$.

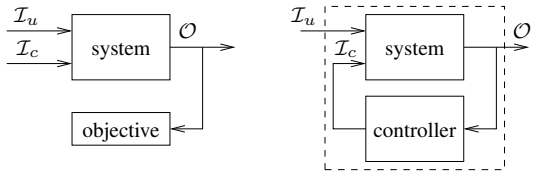

Fig. 1. From uncontrolled system (left) to closed-loop control (right)

As illustrated in Figure 1, the objective is expressed in terms of the system's outputs. The controller is obtained automatically from a user specified LTS and *objective*, according to [Marchand et al. (2000)].

2.3 Optimal discrete controller synthesis

It is also possible to consider *weights* assigned to the states and/or inputs/outputs of $\mathcal{P}$, and to specify that some upper or lower bound must never be reached. *Optimal DCS* [Kumar and Garg (1995); Tronci (1996); Sengupta and Lafortune (1998); Marchand and Samaan (2000)] can then be used to control transitions so as to minimize/maximize, *in one step*, some function w.r.t. these weights; i.e., *go only to next states with optimal weight* [Girault and Rutten (2004)].

In this paper, the problem of finding an optimal strategy *on a path* is considered. For a system M together with a complete cost function C, the optimal strategy leads M from its initial state q_0 to some final states Q_f. The execution path E which is followed from q_0 to $q_f \in Q_f$ must have the lowest execution cost that can be guaranteed. Indeed, if one or more minimal cost paths exist, it cannot be guaranteed that they are systematically followed. Uncontrollable events might drive the system into "bad" cost states such that a global minimum is not reached.

In terms of the concrete transition systems seen previously, we define $C(q)$ be a cost function mapping each potential state of an LTS to a strictly positive integer cost: $C : \mathcal{Q} \rightarrow N$. The *execution cost* EC of a path of length k starting at state q_1, $E(q_1) = (q_1, \ldots, q_k)$ is obtained by adding the static costs of the states in E.

Bellman's algorithm [Bellman (1957)] takes into account this aspect. It operates following two steps. At the first step, each state q of the transition system M is mapped to the best execution cost achievable to reach Q_f, by taking into account the *worse-case* of uncontrolable inputs. Note that this cost value is not necessarily the minimal execution cost achievable. If such an execution path does not exist, then the best cost achievable equals $+\infty$. Let $W : B^p \rightarrow N$ be the mapping function. W is defined as the greatest fixed point of the following recurrent equations:

$$W^0(q) = \begin{cases} 0 & \text{iff } q \in Q_f \\ +\infty & \text{otherwise} \end{cases}$$

$$W^i(q) = \min \begin{cases} W^{i-1}(q) \\ \max\limits_{\mathcal{I}_u} \min\limits_{\mathcal{I}_c} C(q) + W^{i-1}(\delta(q, i_u, i_c)) \end{cases}$$

Download English Version:

https://daneshyari.com/en/article/723941

Download Persian Version:

https://daneshyari.com/article/723941

Daneshyari.com