An Evaluation of the TCT Tool for the Synthesis of Controllers of Discrete Event Systems

Christopher M. Enright
Michel Barbeau

Département de mathématiques et d'informatique
Université de Sherbrooke
Sherbrooke (Québec) CANADA J1K 2R1

Abstract ——— Increasingly, software engineering professionals are attempting to integrate automated systems development into their projects. One well-known instance is YACC, which, when given the specification of a language, can produce that language's parser. In this instance, we tested an automated development tool, TCT. Specifically, we evaluated its ability to synthesize the controller of a discrete event system (DES), namely, a mine drainage system.

Introduction

This paper comprises three sections. The first section is concerned with the TCT interface, with emphasis on its inherent strengths and weaknesses. The second section is a detailed description of the DES problem model used as input for the TCT tool tests. The third section follows a sequential development of the controller synthesis algorithm as proposed by Ramadge and Wonham [2,3], with respect to a TCT approach.

The TCT Tool and Interface

The TCT tool is a software program implemented on the IBM PC DOS platform, and it is written in Borland's Turbo Pascal 6.0 (TP6). The use of TP6 could possibly lead to a few problem areas. For example, the use of TP6 reserved words as DES file names could potentially result in data corruption troubles. Aware of this possibility, the authors, have expressed their concerns in the "readme" text accompanying the TCT tool. Also available is a MS-Windows 3.1 version, which does not significantly differ from its DOS predecessor. As well, the overall nature and the text based look and feel of the system are carried through to the windows product. The lack of a graphical metaphor in the representation of a DES in the MS-Windows product is an unfortunate oversight on the part of the TCT authors, as well as a dissipation of the power and resources of Windows.

With either version, the tool's structure is composed of various sub commands (procedures), each one capable of carrying out a transformation on an input DES. The different commands and their functional description can be found in [2], or a brief summary can be found in the "Introduction to TCT" option of the "MAIN OPTIONS" menu.

The most evident drawback of the tool is its textual interface used for both menu selections and representation of a DES, its states and transitions. For example, the "main control" menu offers the user a choice of an "Introduction to TCT" or to perform a "RESET DES file Directory Path" before getting to the TCT PROCEDURES menu. It would have been more useful to put these options directly in the TCT PROCEDURES menu. One reason for this is that it would allow direct access to the "help" information included in the "Introduction to TCT" option during DES manipulation, a time when this information is needed most (since there is no other "help" system available). Another reason, is that the inclusion of the "RESET .DES file directory path" option in the "TCT Procedures" menu would allow for direct reallocation of the file structures during a TCT session.

The "Main option" screen and all subsequent screens are based on a menuing system where the user must enter a letter or number associated with the option he/she wishes. This letter/number option is not necessarily associated with the choice itself, thus it is not mnemonic. For example, the "1: Create" option, the "1" being the choice tag and not "C"; although with other options such as "M: Minstate", the "M" is both the tag and the first letter in the choice title (as it should be). This inconsistency can lead to a confused user. In this age of GUI's, a control menu should, at the very least, use a bar that moves through the options with cursor control.

The "TCT PROCEDURES" menu incorporates all the DES manipulation routines of the tool, as well as a "# Kbytes available on heap: XXX" message that does not seem to have any relation to the DES manipulation. We tried to track the variations in the heap size when performing system routines and found that there seemed to be no apparent relation to the system function.

In order for the reader to understand the functionality of TCT, we will use the "1: Create" procedure to illustrate the main points of the system. Having chosen the "1: Create" option, the user must name the DES to create (remembering not to use TP6 reserved words as names). Once the name is accepted by hitting the carriage return key, there is no way of changing it except to exit to DOS and use the rename command, an action that proves to be quite cumbersome. The next step in creating a DES is to enter its "size". We, as users, are not given maximum value for this parameter anywhere, and due to the lack of an interactive help system, we can only guess its limits. The DES size is defined in terms of positive integers to some N maximum. Once the size has been entered, the next step is to enter "marker states" for the DES. The marker states are entered as a table of positive integer values with negative one (-1) as the terminating input. Again, a rough management structure. A marker state can be

0-7803-1443-3/93 $3.00

defined as any integer from zero to the total number of states minus one, because the system numbers the states 0 to N-1, N being the total number of states. Once the marker states are entered, the only way to modify them is to finish the "1: Create" option, return to the "PROCEDURES MENU" and use the "E: Edit" option. The next step in creating a DES model through the "1: Create" option is to enter the TCT transition representations. This takes the form of a positive integer triplet, with the first input being the exit state (which must exist in the DES), the second being a transition tag, with uncontrollable events being tagged as even positive integers (0,2,4,...) and controllable events being tagged as odd positive integers (1,3,5,...), and the third input being the entrance state (which must also by definition be within the DES). An example of this is "0 2 1" where "0" is the initial (exit) state tag, "2" is the transition tag (which is "even", therefore it is controllable) and, "1" is the resultant (entrance) state. As with the previous options, the list of marker states is unalterable once entered, except through the external "E: Edit" option. To terminate, a negative one (-1) in the first "exit state" column is required. The system then saves the DES created and requires a carriage return to get back to the "TCT PROCEDURES" menu. If an error has been made at any point while creating the DES, one must use the "E: Edit" option and go through the same cumbersome structures to correct the inaccuracy. Any time during the construction of the DES, it is possible to exit the "1: Create" option by using the "Escape" key; although this alternative is destructive and invalidates any of the data that had been input.

Since the "1: Create" option is essentially representative of all the TCT procedures, its brief description gives us a detailed look at the interface and command structures used throughout the tool.

THE MODEL

The mine drainage system [1] has four modules, namely a methane monitor, a pump, a water level detector, and a controller (Figure 1). The **methane monitor** has two states: "Safe" (methane concentration low) and "UnSafe" (methane concentration high). Its task is to inform the controller when the concentration of methane has reached a critical level.

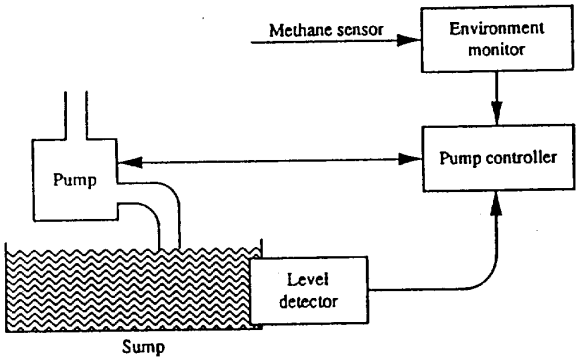

Figure 1 : Mine Drainage System

The pump also has two states: "Idle" (not pumping water) and "Working" (pumping water). Its task is to evacuate water from the mine when signalled by the controller. The **water level detector** consists of five states, "Undefined," "Low," "High," "Low Reset," and "High Reset," its task is to monitor the water level in the mine. The controller regulates the pump and receives signals from the water level detector and methane monitor and reacts accordingly. That is, if the water level is "High" and the methane concentration is "Safe", then the pump must be activated. During all other conditions the pump must be "Idle."

THE PROBLEM

The behaviours of the methane monitor, the water level detector, and the pump are known (Figure 2).

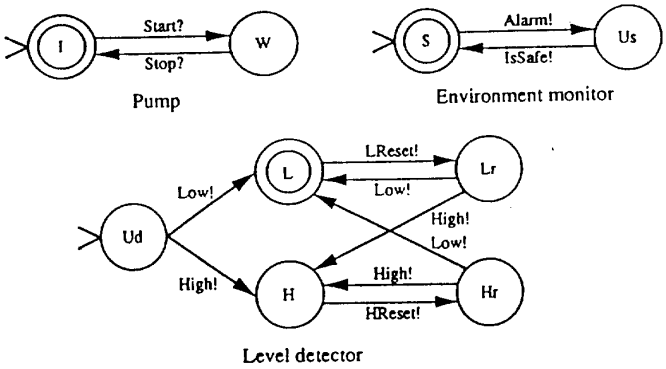

Figure 2 : Behaviours of Known Components

The problem we will address is the construction of the language describing the controller's behaviour. This task will be performed using the synthesis algorithm described in [3] and the TCT tool.

THE TCT APPROACH

Hereafter we will outline the controller synthesis approach [3] and the functionality of the TCT tool.

In order to use the synthesis algorithm we must provide three input parameters. First, "G" the plant DES and its language "L(G)". Second "L" the legal sublanguage of "L(G)" and finally, "M" a function masking non observable events. The legal sublanguage "L" is obtained from "L(G)" by imposing the security constraints stated above and described in [1]. It is required that the behaviour of "G" under the supervision of the controller be included in "L".

The TCT Derivation of the Mine System Model ("G")

In order to form the combined DES of the mine drainage control system, each of the three modules, the behaviours of which were known, were modeled separately using the TCT "1: Create" option and then joined to create the combined DES. This was accomplished by using the TCT "sync" ("4: Sync") procedure, which is designed to take two input DESs and combine them to form the reachable synchronous product by "synchronizing" and "anding" their respective states and transitions to form a third

(output product) DES.

Using TCT to produce the language of the mine drainage system, the user is forced to synchronize the three DESs in a two-step fashion. The first step is to combine ("sync") the "PUMP" DES and the "MONITOR" DES to form an intermediate "PUMPMON" DES, then that product is "synced" with the level "DETECTOR" DES to form the mine drainage system DES "G". The three modules of the system combine to form a DES of 20 states and 72 transitions. This complex two-step procedure (Figure 3) was necessary because TCT cannot deal with more than two input DESs during any of its procedures. This limitation results in a decrease in the system's efficiency and usability.

PUMPMON = Sync (PUMP, MONITOR)
G = Sync (PUMPMON, DETECTOR)

Figure 3 : Production of the mine system model

Once "G" was arrived at through TCT, the increased states and transitions generated by TCT had to be verified correct. This was accomplished by mapping the TCT product to a previous table that had been created manually [1]. The TCT generated product had reacted correctly, since its product was found to be accurate.

Construction of the Legal Language (L)

The next step is to eliminate from "G" the states and transitions determined to be illegal according to the security rules formulated in [1], with the result being "L", the legal language of the DES. The TCT tool's capacity for automatic translation of security and liveliness properties into a legal language are limited. In fact, the only way of automatically incorporating these properties into a TCT model is to use the "Mutex" (6: Mutex) option, which forces compliance with mutual exclusion properties of a given legal language. The "Mutex" command could have been used in this example, but due to the relative simplicity of the security properties used and the difficulties with the TCT interface, we proceeded manually removing the illegal states and transitions from "G" with the "E: Edit" function.

Calculation of the Largest Recognizable Sublanguage

The synthesis algorithm [3] requires computation of the largest recognizable sublanguage "L_r" of "L", mathematically defined as:

$$L_r = L - M^{-1}M(L(G)-L)$$

To compute "L_r" with the TCT tool, we break the equation down into sub-equations, starting with the rightmost section.

1) We calculate the set of illegal words of the plant, i.e. L(G) - L. TCT has no subtraction operation, so we must carry out the operation in two steps as follows :

D1 = COMPLEMENT (L) ; Σ^* - L
D2 = MEET (G,D1) ;L(G)∩D1

(**COMPLEMENT** forms the marked language complementary to the input DES and a list of event labels.
MEET forms the reachable cartesian product of the two input DESs.)

2) We mask the illegal words with respect to the plant's non-observable events, i.e. computation of M(L(G)-L), in effect describing how the illegal words are observed.

D3 = PROJECT (D2, {8,18})

(**PROJECT** forms the closed and marked language of the input DES with listed events erased.)

3) We take the inverse of the second step's result, constructing the set of all legal and illegal words observed as illegal words (i.e. $M^{-1}(M(L(G)-L))$).

D4 = SELFLOOP (D3, {8,18})

(**SELFLOOP** augments the input DES by attaching selfloops at each event in the list.)

4) This step eliminates from "L" all legal words that have illegal counterparts observed in the same manner, i.e. $L-(M^{-1}(M(L(G)-L)))$.

D5 = COMPLEMENT (D4)
D6 = MEET (L,D5)

L(D6) is the largest recognizable sublanguage of "L".

Synthesis of the Controller

1) We calculate the model of the system as seen by the controller by masking "L(G)" with respect to the non- observable events.

D7 = PROJECT(G,{8,18}) ; L(D7)=M(L(G))

2) We also mask "L_r" (D6) with respect to the non-observable events, thereby deriving the projection of the largest recognizable sublanguage of L.

D8 = PROJECT(D6,{8,18}) ; D8=M(L(D6))

3) We then take these results and generate the controller for the plant, "G".

D9 = SUPCON(D7,D8)

(SUPCON, for the first input parameter, forms a trim recognizer for the supremal controllable sublanguage of the marked legal language of the second input parameter, which in turn is the controller for the first input parameter.)

L(D9) is the plant controller automaton. We verified the results manually using document [1], TCT performed correctly.

This entire multi-step algorithm has been incorporated into the most recent version of TCT in the form of a single operation, the "R1: SUPNORM" option. We have taken the trouble to describe

the detailed process to allow the reader to view the basic operations of TCT relative to the synthesis of controllers. The same results are achieved if one uses the "SUPNORM" option to solve this example.

CONCLUSION

The TCT program is a necessary first step in the quest for an automated product that can synthesize basic software engineering activities; however, it is certainly not the final solution. The authors have invested most of their time and effort on the functional and procedural aspects of the internal design process, to the detriment of a very important aspect of software development, the external design architecture. The TCT product's usability and effectiveness are reduced due to a cluster of interface design problems such as the use of textual interface structures, non-editable input fields, negative one (-1) terminating flags and integer based flags, as in the case of transition labels. Even though the TCT interface is scant, its internal structures appear to be well engineered, powerful, and they accomplish what they are programmed for and it is still a powerful and useful tool in the modelling of a DES. A simple addition, such as the use of alphanumeric identification labels would be a considerable improvement in promoting a greater understanding of the modeled problem.

The MS-Windows version of the tool vanquished a great many hopes by continuing to incorporate the same control and labelling structures of its DOS predecessor. The fact that the TCT authors did not embrace the GUI paradigm to model a DES would suggest that they have focused their effort elsewhere, most certainly on the internal structures of the program. This dedication of effort to the DES manipulation procedures gives us insight into the complexity of code necessary to automate the DES manipulation algorithms. That said, the authors would be well advised to take a step into the larger graphical universe, thereby, for the most part, remedying TCT interface limitations and increasing the system's ease of use. A system is only as good as the user's ability to use it, after all; the best programming in the world could be rendered useless by masking it with a cumbersome and unwieldy interface.

Direct improvements to the conceptual base of TCT as an automated software synthesis system would be twofold. First, the addition of a procedure specifically suited to the operation of deriving the legal sublanguage of a language with respect to liveliness and security properties. An example of the potential benefit of this addition would be the automatic removal of states and transitions deemed illegal with respect to certain properties, such as presented in [1].

Secondly, a fully graphical interface directly modelling DES structures would be a great improvement. The ability to directly edit graphical DES models would greatly improve the user's comprehension of the system and increase his/her effectiveness with using the tool. The graphical interface should also allow for a host of user friendly additions such as the ability to name, tag, and label, alphanumerically, graphical images, that represent states and transitions. As well, the addition of the GUI paradigm would allow the level of complexity of the modeled system (DES) to increase considerably.

It is well documented that "a picture is worth a thousand words (or numbers)", so a graphically based software synthesis program would be a truly realistic, and understandable model of the DES system in question. This evolution to a graphical system would be the next great step towards achieving our ultimate goal, of automation in the synthesis of software projects.

ACKNOWLEDGEMENTS

We would like to thank Professor Murray Wonham of the University of Toronto Electrical Engineering faculty for the use of the TCT tool. Without his willingness to share his work, this article would not have been possible.

As well, we would like to thank Dr. S.M. Enright and L.R. Ball,for their unending encouragement and Dr. R. Sutherland and M.C. Hopps for their support and syntactical contributions.

REFERENCES

[1] M. Barbeau, G. Custeau, R. St-Denis, "A Synthesis- Based Solution to a Mine Drainage Control System Problem," Technical Report no. 106, Université de Sherbrooke, Département de mathématiques et d'informatique, 1992.

[2] W.M. Wonham *Notes on Control of Discrete-Event Systems.* Toronto, University of Toronto, Department of Electrical Engineering, 1992.

[3] P.J. Ramadge and W.M. Wonham "Supervisory control of a class of discrete event processes," SIAM Journal of Control and Optimization, 25 (10 1987), 206-230.