SyncCharts : un formalisme graphique pour systèmes réactifs complexes

Charles ANDRÉ
Laboratoire Informatique, Signaux, Systèmes (I3S)
Université de Nice-Sophia Antipolis / CNRS
41, bd Napoléon III — F – 06041 NICE Cedex
e-mail: andre@unice.fr

ABSTRACT

Les systèmes réactifs nécessitent communication, parallélisme et préemption. Peu de modèles supportent ces trois concepts. L'approche synchrone, qui autorise leur coexistence, est d'un intérêt majeur pour les systèmes réactifs.

Dans cette présentation nous introduisons un modèle graphique, appelé SYNCCHARTS, qui s'appuie sur le paradigme synchrone. Nous illustrons son emploi sur un système simple mais qui rassemble les principales caractéristiques des systèmes réactifs. Nous introduisons également l'environnement de développement associé aux SYNCCHARTS.

Mots clefs : modèle graphique, systèmes réactifs, paradigme synchrone, SYNCCHARTS.

1 INTRODUCTION

Un *système réactif* est en interaction permanente avec son environnement. Il met généralement en œuvre du parallélisme et il est souvent constitué de plusieurs sous-systèmes interagissants. Le comportement observé résulte de la coopération de ces sous systèmes. La communication est essentielle pour assurer cette coopération ; ceci explique que très souvent les systèmes réactifs sont vus comme des systèmes communicants. Cette vision peut occulter les *préemptions* qui sont au moins aussi importantes dans les systèmes réactifs. La préemption, dans le sens général que nous lui donnons, exprime le fait qu'un agent peut voir son activité arrêtée (avortement) ou suspendue (suspension) par d'autres agents. Peu de modèles sont capables de traiter proprement, c'est à dire avec une sémantique rigoureuse, la coexistence du parallélisme et de la préemption. L'*approche synchrone* des systèmes réactifs a été introduite pour répondre, entre autre, à ce problème. Différentes formes de préemptions ont été définies par G. Berry, dans le cadre synchrone [1].

Les lecteurs intéressés par l'approche synchrone peuvent se référer aux articles d'introduction à cette approche publiés dans les proceedings des IEEE [2] en 1991. La lecture du livre de N. Halbwachs [3] est également recommandée. Plus récemment, nous avons publié une synthèse sur cette approche, en dégageant les hypothèses sous-jacentes, avec un point de vue "automaticien" [4]; nous y avons également introduit le modèle SYNCCHARTS qui se place résolument dans le cadre de cette approche.

Dans cette présentation, nous nous contentons d'illustrer l'usage des SYNCCHARTS, en dégageant leurs principales caractéristiques et en indiquant les supports logiciels existants, ainsi que les développements en cours.

SYNCCHARTS est l'acronyme de *Synchronous Charts*. Ils sont fortement inspirés des *Statecharts* de D. Harel [5], mais ils offrent une plus grande richesse au niveau des préemptions et ils bénéficient d'une sémantique plus stricte basée sur celle du langage ESTEREL. Les SYNCCHARTS sont également à rapprocher d'ARGOS [6], par leur sémantique. Bien sûr, en tant qu'automaticien, le GRAFCET [7] (Sequential Function Charts) a également été un modèle d'inspiration. Les SYNCCHARTS ne sont toutefois pas limités aux variables booléennes : ils s'adressent aux systèmes réactifs, en général.

2 EXEMPLE : CHIEN DE GARDE

Nous allons modéliser un chien de garde que l'on peut armer, désarmer ou geler.

2.1 État – Transition

De nombreux modèles conceptuels adoptent une description état/transition. C'est le cas des SYNCCHARTS qui ont à la fois les caractéristiques des machines de Moore et de Mealy.

Dans la figure 1, les états ne sont pas explicitement nommés car ce sont les entrées et sorties qui sont primordiales dans les systèmes réactifs. Pour faciliter les explications, appelons $S0$ l'état situé à gauche et $S1$ celui situé à droite.

$S0$ est un état *initial*, ce qui est matérialisé par un arc venant d'un *connecteur* (rond noir) et pointant sur un état. Dans cet exemple, aucune action (sortie) n'est associée à $S0$. Par contre, quand $S1$ est activée, on émet le signal `b0` (à la façon du GRAFCET). `T` est l'entrée qui déclenche la transition de l'état $S0$ à $S1$. `T` peut être interprété comme un ordre de basculement. La transition de $S1$ à $S0$ est également déclenchée par `T`, mais cette transition s'accompagne de l'émission du signal `c1`, comme dans une machine de Mealy. `c1` est interprété comme une retenue.

Le petit rond situé à l'origine des arcs n'est pas une fantaisie graphique. Il est lié à la préemption, il sera justifié plus loin.

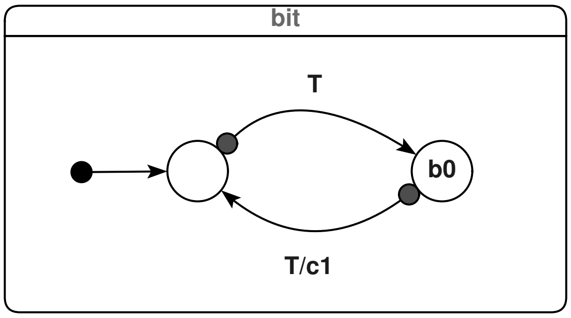

FIG. 1 - SYNCCHARTS *d'un compteur binaire 1 bit.*

2.2 Parallélisme

Les graphes d'état sont bien connus pour leur caractère explosif en présence d'évolutions simultanées. Les modèles comme les réseaux de Petri réduisent, au niveau de la représentation, cet inconvénient. Les SYNCCHARTS ont la possibilité, directement inspirée des STATECHARTS, de représenter des graphes d'états en parallèle. Les lignes pointillées séparent les différentes composantes parallèles. La figure 2 modélise un compteur binaire à trois bits. Il est constitué de 3 instances du compteur 1 bit. Il y a un état et un seul actif dans chaque composante. Les graphes en parallèle ne sont pas totalement indépendants car leurs évolutions peuvent se synchroniser comme on l'explique dans le paragraphe suivant.

2.3 Communication

Dans une approche synchrone, toutes les composantes perçoivent simultanément les mêmes informations. Ceci résulte de l'hypothèse de *diffusion instan-*

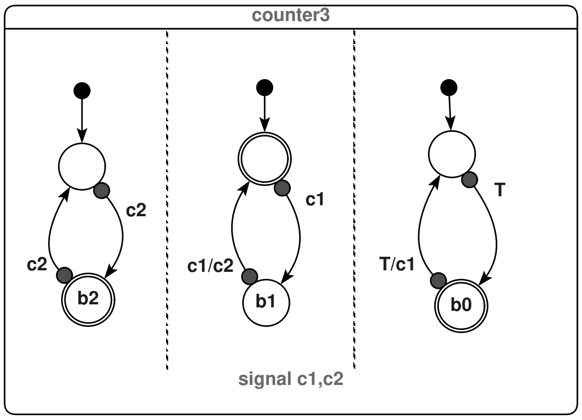

FIG. 2 - SYNCCHARTS *d'un compteur binaire 3-bits.*

tanée. Une conséquence est qu'une émission faite dans une composante parallèle peut très bien déclencher *dans le même instant* une évolution dans une autre composante. C'est le cas des signaux `c1` et `c2` dans le compteur 3-bits. Ces signaux sont *locaux*, c'est à dire qu'ils ne sont ni directemnet observables, ni directement commandables de l'extérieur. Ils ont été introduits pour faciliter la description : ils sont conceptuels mais pas nécessairement physiques. Sous les hypothèses synchrones, le compteur passera de l'état 3 à l'état 4 sans intermédiaire : les 3 bits changent d'état simultanément. Le GRAFCET nous a déjà habitué à ce type d'évolution.

Dans cet exemple, il s'agit de synchronisation : seules les occurrences des signaux sont significatives. Plus généralement les signaux peuvent être porteur de valeur (analogue des signaux valués d'ESTEREL). Les signaux sont donc un moyen de *communication instantanée.*

2.4 Hiérarchie

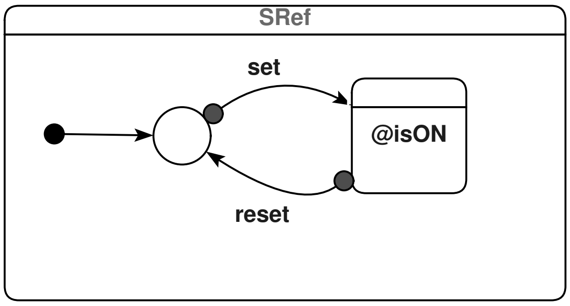

FIG. 3 - SYNCCHARTS *d'une SR raffinée.*

La représentation "plate" des graphes d'état est un inconvénient. Les STATECHARTS et ARGOS autorisent des automates hiérarchisés. Cette possibilité a été reprise dans les SYNCCHARTS. On peut *raffiner* un état.

Graphiquement, les ellipses sont utilisées pour les états non raffinés, alors que les rectangles à coins arrondis modélisent les états raffinés ou *macro-états*. Noter le caractère récursif de cette description : un syncChart lui-même est un macro-état.

Dans la figure 3, l'état `ON` de la bascule SR a été raffiné. Le raffinement peut être directement représenté dans le macro-état ou être dessiné par ailleur (cas de la figure 3 qui renvoie à `isON`, le symbole @ indiquant cette indirection).

2.5 Préemption

La préemption peut tuer un agent ou bien le suspendre temporairement. La *suspension* est forcée tant que le signal de suspension est présent. Dans la figure 4, le macro-état `counter3` est suspendu chaque fois que le signal `inhib` est présent. Il n'y a aucune évolution interne dans un macro-état lorsque celui-ci est suspendu ; on obtient un véritable "figeage". La suspension est représentée par un arc venant d'un connecteur (point noir) et se terminant sur un état ou un macro-état par un rond rouge (en grisé dans le texte).

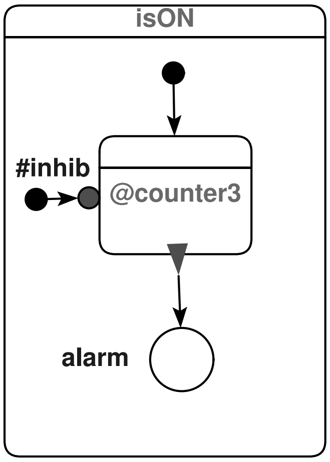

FIG. 4 - SYNCCHARTS *d'une SR raffinée.*

L'*avortement*, par contre, met définitevement fin à l'exécution d'un agent. On distingue deux formes d'avortement [1] appelées respectivement faible et forte. Cette distinction est propre à l'approche synchrone qui s'appuie sur une notion précise d'*instant*. Un arc d'avortement a pour origine l'état ou le macro-état à préempter et à pour terminaison l'état ou macro-état suivant. L'*avortement fort* interdit toute exécution de l'agent préempté, dans l'instant de préemption. Il est représenté par un arc ayant comme origine un rond rouge (en grisé sur les figures). L'*avortement faible* laisse au contraire terminer l'instant avant la destruction effective ; l'agent tué peut, en fait, exprimer ses "dernières volontés". Graphiquement un arc d'avortement simple est un arc ordinaire.

Pour illustrer la différence entre les deux types d'avortement reprenons la figure 1 en supposant que dans l'état $S0$ on émette `nb0`. Si initialement nous sommes dans l'état $S0$ et qu'à l'instant k arrive l'événement déclencheur `T`, on a les comportements suivants :

instant	$\cdots$	$k-1$	k	$k+1$	$\cdots$
Av. faible	$nb0$	$nb0$	$\{nb0, b0\}$	$b0$	$\cdots$
Av. fort	$nb0$	$nb0$	$b0$	$b0$	$\cdots$

Causalité : Notons qu'un avortement faible peut être provoqué par l'émission du signal déclencheur par l'agent lui-même (suicide). Cette possibilité est exclue pour un avortement fort puisqu'il conduira à un comportement absurde : un agent constate qu'il doit se détruire en éxecutant ce qu'il n'aurait pas dû exécuter! Cette situation est un exemple de *cycle de causalité* tel qu'on en rencontre dans une approche synchrone impérative.

Futur et futur strict : par défaut on attend toujours une occurrence strictement future des signaux déclencheurs. Si on veut tenir compte d'un présence éventuelle dans l'instant courant on utilise un modifieur, noté par le symbole `#`, et qu'on lit "immédiat". La suspension par le signal `inhib` dans la figure 4 est du type immédiate.

Terminaison normale : généralement on ne quitte un état que par un avortement. Dans certains cas il peut y avoir une terminaison spontanée dite *terminaison normale*. Pour représenter cette possibilité, nous utilisons des arcs dont l'origine est un triangle rouge venant de l'intérieur d'un macro-état. Les SYNCCHARTS se distinguent des STATECHARTS et d'ARGOS en utilisant des états dits *finals* (représentés en traits doubles). Lorsque *toutes* les composantes parallèles d'un macro-état sont dans un état final, alors il y a instantanément terminaison normale du macro-état. Le compteur binaire figure 2 illustre cette possibilté : le macro-état `counter3` termine normalement exactement lors de la cinquième occurrence de `T` qui suit l'entrée dans ce macro-état.

2.6 Comportement

Une vue en "boîtes transparentes" du SYNCCHARTS décrivant le comportement du système réactif "chien de garde" est donnée figure 5. Le chien de garde est armé par `set` et désarmé par `reset`. Une fois armé, si 5 occurrences du signal `T` ont lieu (en l'absence du signal `inhib`) alors on déclenche une alarme (signal `alarm`). Ce signal ne peut être interrompu que par une occurrence de `reset`. À tout instant, `reset` peut faire avorter le comptage, c'est à dire désarmer le chien de garde.

Un exemple plus conséquent (régulateur de vitesse de croisière) a été présenté dans le papier introductif aux SyncCharts [4] et repris de façon plus détaillée dans un rapport technique [9]. On y retrouve parallèlisme, communications instantanées et préemptions.

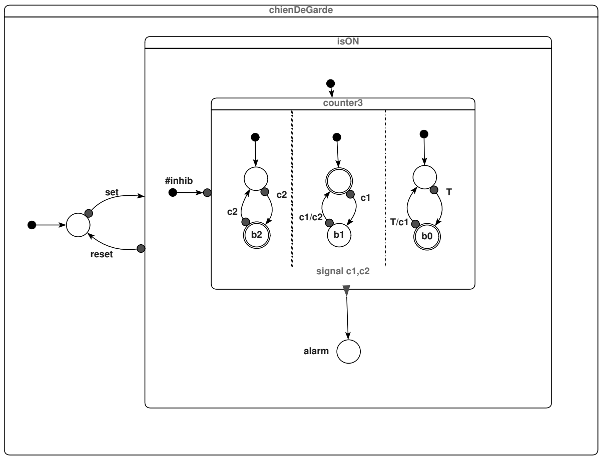

Fig. 5 - SyncCharts *de l'application "Chien de Garde".*

3 SYNCCHARTS

Toutes les figures précédentes ont été produites par un éditeur de SyncCharts.

3.1 Syntaxe

La figure 6 résume les éléments graphiques qui composent les SyncCharts. Ils ont été illustrés dans l'exemple "chien de garde". Les SyncCharts ont été définis avec des vocables anglais. Les éléments de base sont *star*, *constellation* et *firmament* (ou *macro-state*).

Une étoile (star, Figure 6.B) est plus qu'un état : c'est un état et les façons de quitter cet état. Les arcs d'avortement issus d'un état constituent les "rayons" de l'étoile.

Une constellation (Figure 6.C) est un ensemble d'étoiles tel qu'une et une seule étoile est active à l'intérieur d'une constellation.

Un macro-état (Figure 6.A) est constitué de 1 ou plusieurs constellations en parallèle.

Les arcs sont de deux types :

- les arcs d'avortement, entre deux étoiles ou macro-états,
- les arcs auxiliaires, depuis un connecteur vers une étoile ou un macro-état.

Les arcs sont étiquetés. La forme générale d'une étiquette est :

`trigger[condition]/effect` avec

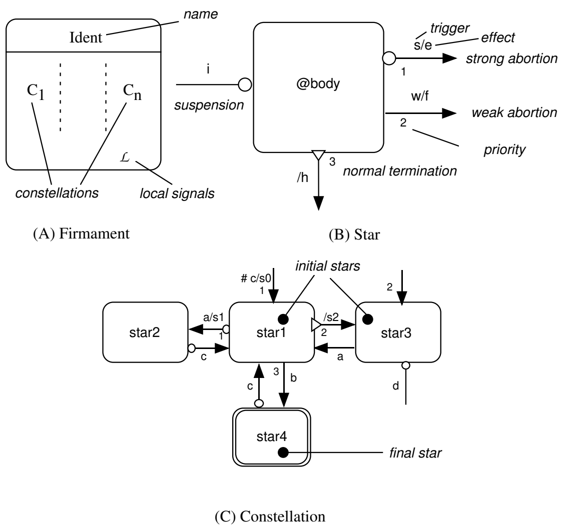

Fig. 6 - SyncCharts*: Éléments de syntaxe*

- `trigger` : une expression booléenne (utilisant les opérateurs `and, or, not` et des parenthèses) portant sur la présence des signaux,
- `condition` : un prédicat pouvant faire intervenir les valeurs de signaux,
- `effect` : une liste de signaux, éventuellement valués, à émettre lors de la transition.

Ces différents champs sont optionnels. Le défaut pour le trigger est le signal `tick`, un signal prédéfini présent à chaque instant. Le trigger peut avoir un effet immédiat si il est préfixé par le symbole `#`.

`set; T/c1; T [?T > nextTime]`
sont des exemples d'étiquettes. Les deux premières ont déjà été vues. La troisième déclenche la transition sur l'occurrence du signal `T` si la valeur de ce signal (disons de type `time`) est supérieure à la valeur de la constante `nextTime` du même type.

Un SyncCharts a un comportement entièrement déterministe. Dans le cas où il existe plusieurs façons de quitter une étoile, on doit donner un ordre d'évaluation. Les entiers $1, 2, \cdots, n$ situés près de l'origine d'un arc précisent la priorité de celui-ci (le plus petit entier est le plus prioritaire).

3.2 Sémantique

La sémantique des SyncCharts est définie en termes d'une algèbre de processus synchrones. Cette algèbre est très proche de celle définie par G. Berry pour le langage Esterel [1]. Ceci nous a permis de donner une traduction *systématique et rigoureuse* des SyncCharts en programmes Esterel. Les détails concernant la sémantique sortent du cadre de cette présentation.

Une description de la sémantique comportementale des SyncCharts est donnée dans le papier d'introduction aux SyncCharts [4]. Un rapport technique, beaucoup plus détaillé explique cette sémantique et la traduction en Esterel [8].

Remarque : Donnons un exemple pour illustrer le fait que nous avons recherché une forte adéquation entre syntaxe et sémantique. Sémantiquement, l'avortement fort est équivalent à un avortement faible par le trigger *et* une suspension, au même instant, par ce même trigger. L'avortement simple est représenté par un arc, la suspension par un petit rond rouge adjacent à l'étoile à suspendre. La préemption forte, sémantiquement combinaison des deux, est syntaxiquement un hybride : un arc dont l'origine est un petit rond rouge.

4 ENVIRONNEMENT DE PROGRAMMATION

Un environnement de programmation a été développé pour les SyncCharts. Il comprend un éditeur graphique et un compilateur vers le langage Esterel. Il intègre également les appels au compilateur Esterel, à un compilateur C et au simulateur xes qui permet l'exécution interactive de programmes Esterel avec remontée dans le source.

L'éditeur/compilateur est écrit en Tcl-Tk [10].

Le compilateur Esterel doit être de version 5.01 ou supérieure, le processeur `sccausal` de la chaîne de compilation Esterel est également indispensable.

Le statut actuel de l'éditeur/compilateur de SyncCharts est celui d'un prototype. Il nous a permis des expérimentations sur des exemples d'école. Un exemple plus conséquent (un dizaine de niveaux et une centaine d'étoiles ou macro-états) a pu être dessiné et compilé, mais au prix d'une grande patience, car dans sa version 1, qui est opérationnelle, l'éditeur tolère très mal les erreurs de l'opérateur.

4.1 Version 1

Écrit avec la version ancienne de Tcl-Tk (Tk3.7), la distribution, en binaire, intègre directement le code et les bibliothèques Tcl-Tk. Le binaire de la version v1.6 n'est disponible que pour stations Sparc tournant sous SUN4OS4.

4.2 Version 2

Cette version, en cours de développement offre des possibilités graphiques nettement améliorés. Elle est programmée dans la nouvelle version de Tcl-Tk(Tk4.1) qui n'est pas pleinement compatible avec les anciennes versions. L'algorithme de compilation est également modifié pour engendré des programmes Esterel de taille plus réduite. Ce compilateur est en cours d'implantation.

Afin que le lecteur puisse se faire une idée de l'interface offerte, nous avons fait des copies d'écran de SyncChartsv2.01, sur lesquelles nous avons rajouté des commentaires.

Panneau principal : Ce panneau donne accès par des menus déroulants aux différents services précisés sur la figure 7.

Arbre des macro-états : Ce panneau visualise l'arbre de l'application et offre des possibiltés de modification de la hiérarchie, par élagage ou greffre de sous-arbres (Figure 8). Il permet également l'accès direct à un macro-état.

Dessin : Chaque macro-état est dessiné dans un canvas indépendant (Figure 9). Une série de boutons permet de choisir les principales fonctions d'édition. Depuis un macro-état on peut remonter vers son ancêtre ou ouvrir un de ses descendants. Le dessin d'un macro-état peut être iconifié ou même retiré à tout moment.

5 Conclusion

Les SyncCharts sont un modèle puissant pour représenter les comportements des systèmes réactifs complexes. Grâce à la chaîne de compilation associée qui permet d'engendrer des programmes Esterel sémantiquement équivalents et de les simuler par xes, SyncCharts permet du prototypage rapide d'applications réactives. Par son caractère graphique, ce modèle est succeptible d'intéresser toute une classe d'utilisateurs réticents à utiliser les langages synchrones. Nous pensons, en particulier l'utiliser dans nos enseignements EEA, comme un "super-Grafcet" et comme introduction au paradigme synchrone.

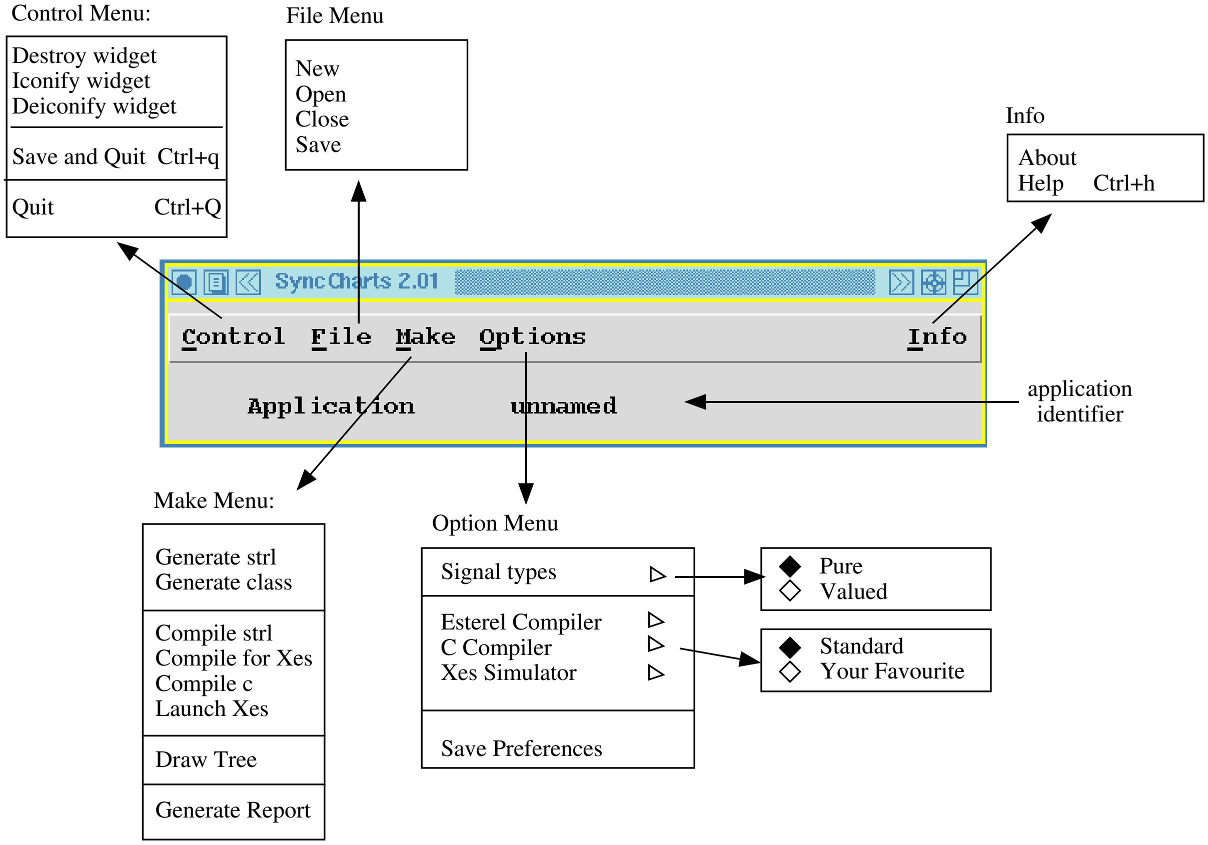

FIG. 7 - SYNCCHARTS : *panneau principal.*

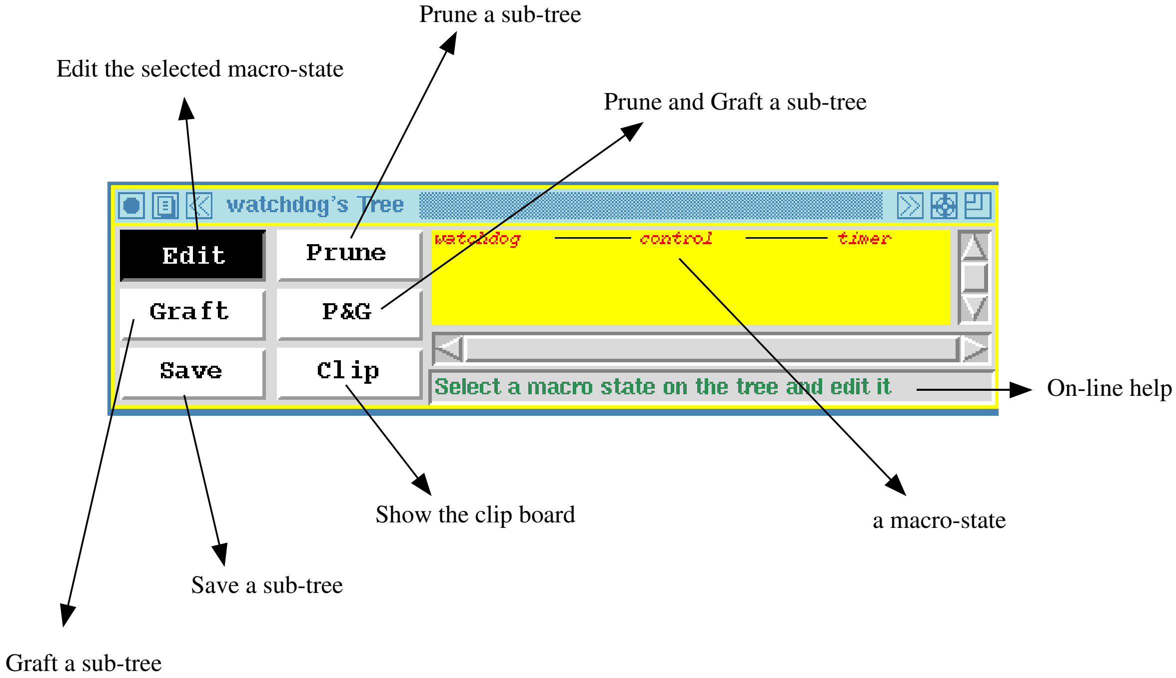

FIG. 8 - SYNCCHARTS : *arbre.*

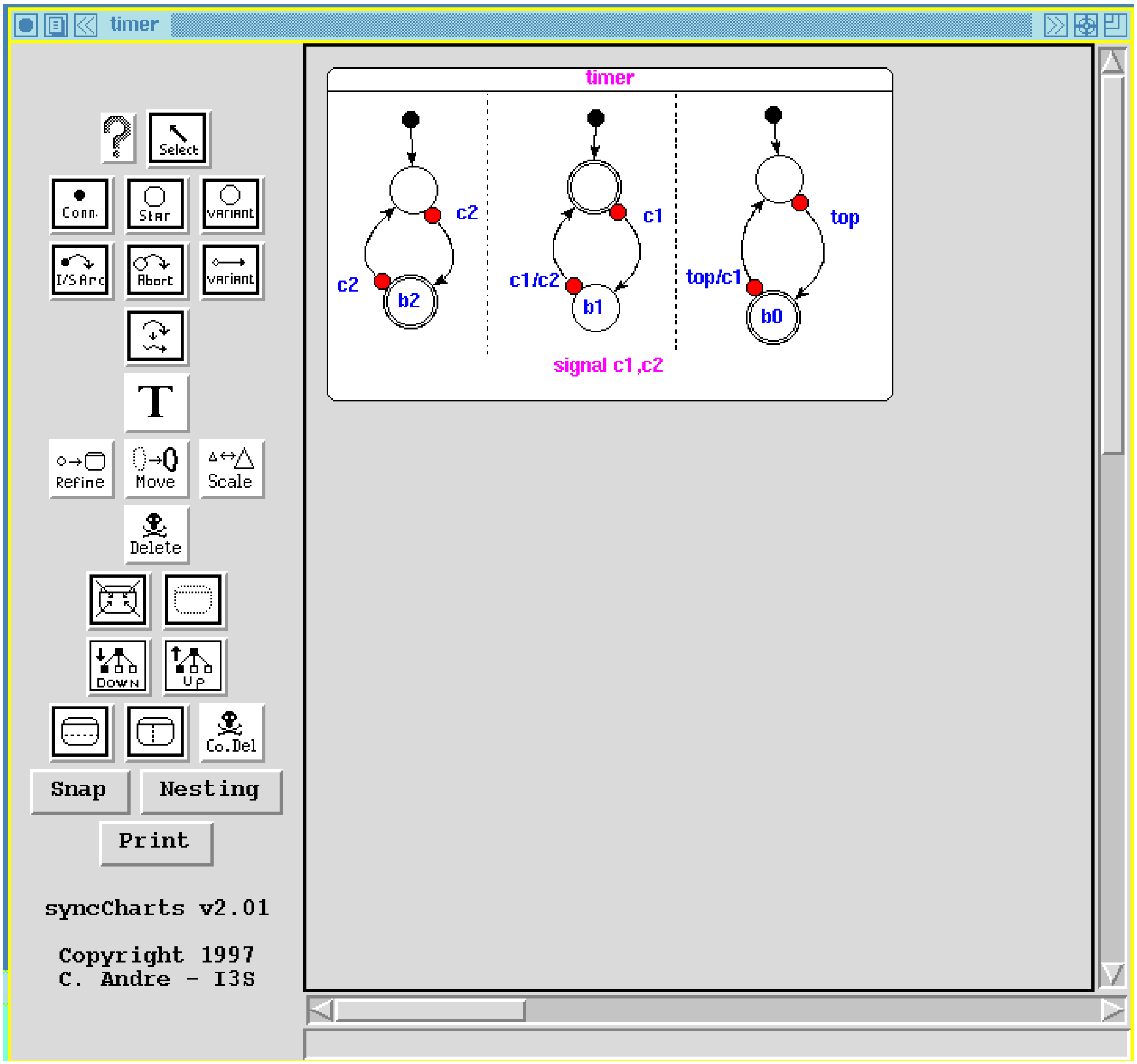

Fig. 9 - SyncCharts : *Édition d'un macro-état.*

Références

[1] G. Berry. Preemption in concurrent systems. *Proc FSTTCS, Lecture notes in Computer Science*, 761:72–93, 1992.

[2] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceeding of the IEEE*, 79(9):1270–1282, September 1991.

[3] N. Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, Amsterdam, 1993.

[4] C. André. Representation and Analysis of Reactive Behaviors: A Synchronous Approach. *IEEE-SMC, Computational Engineering in System Applications, CESA'96*, Lille (F), 1996, pp 19–29.

[5] D. Harel. STATECHARTS: A visual formalism for complex systems. *Science of computer programming*, 8:231–274, 1987.

[6] F. Maraninchi. ARGOS*: un langage graphique pour la conception, la description et la validation des systèmes réactifs.* PhD thesis, Université Joseph Fourier, Grenoble I, Janvier 1990.

[7] IEC, Genève (CH). *Preparation of Function Charts for control systems*, december 1988. International standard IEC 848.

[8] Charles André. Synccharts: a visual representation of reactive behaviors. Technical Report RR 95–52, I3S, Sophia-Antipolis, France, October 1995.

[9] Charles André. Examples of Synccharts. Technical Report RR 96–?, I3S, Sophia-Antipolis, France, February 1996.

[10] J.K. Ousterhout. *Tcl and the Tk Toolkit* Addison-Wesley, Professional Computing Series, 1994.