Projet ECLIPSE

Traducteur Scicos/SynDEx : v2.2.2

Installation et utilisation

Cyril Faure
Cyril.Faure@inria.fr
10 avril 2006

INRIA - Domaine de Voluceau - Rocquencourt B.P.105
78153 Le Chesnay Cedex - France

Chapitre 1

Manuel de référence

1.1 Résumé

Cette version du traducteur prend totalement en compte le contrôle issu de blocs logiques Scicos. Le schéma peut comprendre des blocs multiactivés ainsi que des communications entre blocs d'activations différentes.

A partir d'un schéma Scicos valide, le traducteur :

- génère le graphe d'algorithme SynDEx traduit du schéma Scicos,
- crée un graphe d'architecture monoprocesseur et renseigne les durées d'exécutions de chaque opération aléatoirement de manière à permettre la mise en place rapide de tests sous SynDEx,
- génère le fichier contenant le macro code correspondant aux appels des fonctions contenues dans les librairies Scicos,
- toujours à des fins de tests dans le cadre monoprocesseur, le traducteur génère automatiquement plusieurs fichiers nécessaires à la macro expansion (m4) et compilation du code.

Sont également fournis le code correspondant à des capteurs et actionneurs "de base" ainsi que le code correspondant aux fonctions Scicos générales via le fichier "s2s_common.c".

1.2 Installation

1.2.1 Prérequis

Éléments indispensables concernant la traduction de graphe :

- Scilab version 4+ installé,
- SynDEx version 6+ installé,
- l'archive ScicosToSynDEx.tgz contenant les fichiers du traducteur,

Une fois la traduction effectuée et le code m4 généré sous SynDEx, il est nécessaire de fournir le code correspondant aux blocs Scicos traduits. Un sous-ensemble de ceux-ci se trouve dans l'archive s2s_scicosFiles.tgz. Ces fichiers correspondent simplement au code de chacun des blocs Scicos et sont recopiés tels quels dans le code final pour chaque bloc utilisé. A noter que ces fichiers sont mis à jour par l'équipe de développement Scicos, il est donc recommandé de récupérer régulièrement leur dernière version via le CVS de Scilab.

Egalement, tous les blocs Scicos ne sont pas présents dans l'archive. Certains ne sont pas traduisible dans le contexte "flot de données" de SynDEx (blocs zero-crossing par exemple) alors que d'autres ne se conforment pas aux exigences de codage des blocs pris en compte par le traducteur (i.e, blocs de type 4 codés en C sans appel de fonction externe). Il est à la charge de l'utilisateur de les coder dans le cas où certains blocs ne seraient pas présents dans l'archive[1].

[1] un message d'erreur signale le manque du fichier concerné lors de la macro expansion

1.2.2 Installation

Nous appellerons ici "~Scilab" le chemin où est installé Scilab sur la machine et "~" la racine du répertoire courant de l'utilisateur.

- Créer un répertoire "SCICOS_SYNDEX" dans ~Scilab,
- copier le fichier ScicosToSynDEx.tgz dans ce répertoire,
- décompresser le fichier,
- dans le répertoire ~Scilab doit se trouver le fichier Scilab.star chargé d'exécuter automatiquement des commandes au démarrage de Scilab. Nous voulons que les fonctionnalités du traducteur soient accessible dans Scicos donc :
  ajouter la ligne 'exec SCI/SCICOS_SYNDEX/dot.scilab' à la fin de ce ficher[2].

1.2.3 Problème d'installation

La compression doit conserver le chemin des différents fichiers. Si elle ne le fait pas, voici la structure hiérarchique des fichiers du traducteur :

- Répertoire scilab
  - répertoire "SCICOS_SYNDEX" préalablement créé par l'utilisateur afin d'y décompresser "ScicosToSynDEx.tgz"
    - répertoire "MACROS"
      - macros (fichiers sci) du traducteur
    - fichier "loader.sce"
    - fichier "dot.scilab"

1.3 Fonctionnement du traducteur

1.3.1 Prérequis

Aucun prérequis n'est indispensable dans le sens où l'approche de la conception du traducteur a été d'automatiser au maximum les traitements. Cependant, il est recommandé d'avoir une bonne connaissance de l'utilisation de SynDEx et surtout du processus de macro expansion du code m4 généré par SynDEx afin de pouvoir intervenir sur les détails liés à l'implantation finale du code exécutable tels que l'inclusion de fichiers supplémentaires, l'utilisation de sensors/actuators personnalisés, la modification du nombre d'itérations de la boucle d'algorithme, etc.

1.3.2 Utilisation

Le mode de fonctionnement est similaire à la version précédente :

- on crée sa spécification,
- on crée un superbloc ayant une et une seule entrée d'activation à l'aide de la fonctionnalité "region to superbloc" de Scicos,
- on clique sur "Object ⇒ TO SynDEx" puis sur le superbloc.

Interface utilisateur :

S'entame alors une phase pendant laquelle Scicos précompile le schéma puis rend la main au traducteur qui affiche l'interface suivante :

[2] dot.scilab est contenu dans l'archive décompressée

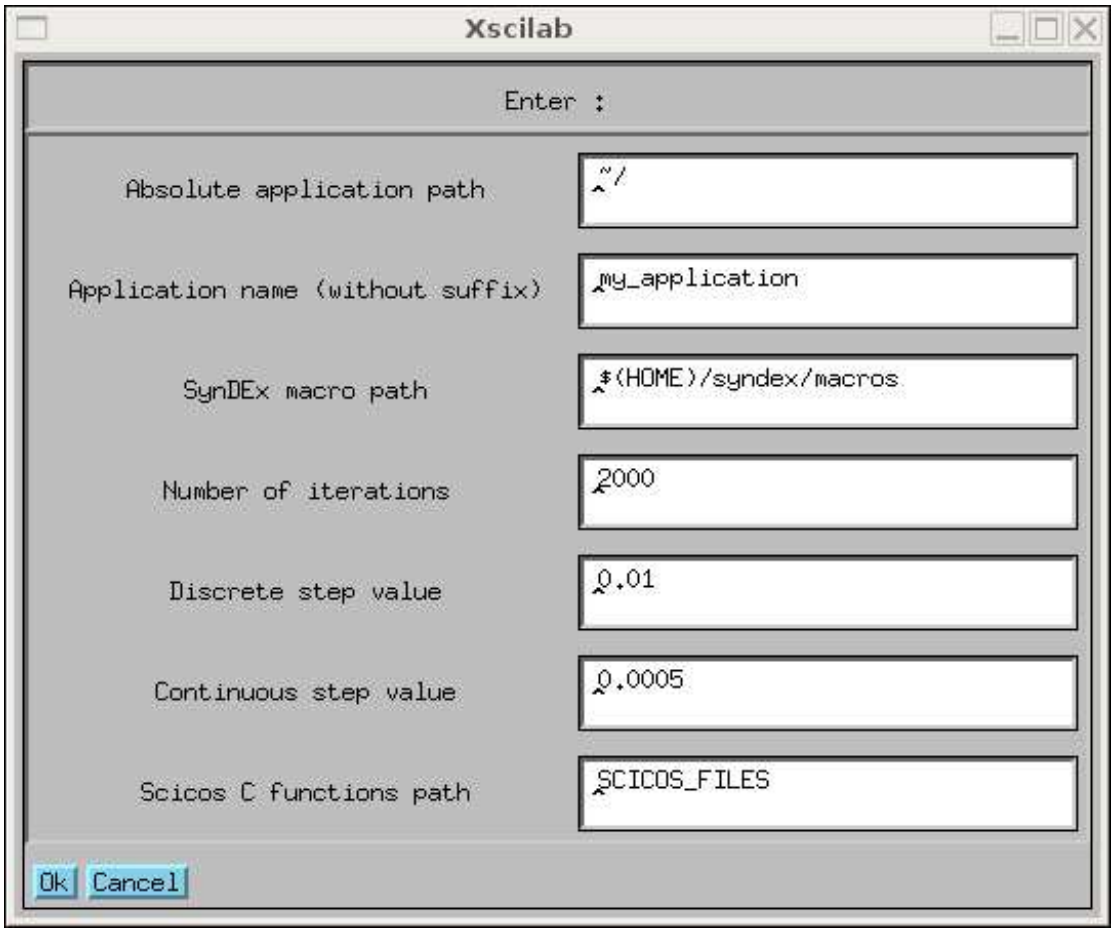

FIG. 1.1 – Interface utilisateur du traducteur

Les champs à remplir sont :
- Absolute application path, est le chemin absolu vers lequel seront créés tous les fichiers issus de la traduction,
- Application name, est le nom de l'application que l'utilisateur souhaite,
- SynDEx macro path, correspond au chemin à partir duquel le mécanisme de macro expansion pourra trouver les macros nécessaires à la génération du code C final,
- Number of iterations, est le nombre de "tour de boucle" qu'effectura l'application finale lors de son exécution,
- Discrete step value, est la valeur du pas discret,
- Continuous step value, est la valeur d'un pas d'intégration. L'important étant le rapport du pas discret sur le pas continu ('équivaux au nombre d'étapes d'intégrations par pas d'éxécution discret),
- Scicos C function path, est le chemin à partir duquel le mécanisme de macro expansion pourra trouver les fichiers C correspondant au code des blocs Scicos.

Fichiers générés :

Soit SAP (Syndex Application Path) le chemin de l'appli dans lequel générer les fichiers et SAN (Syndex Application Name) le nom de l appli. Plusieurs fichiers sont générés dans SAP. Les fichiers principaux sont :
- SAN.sdx : le fichier lisible par syndex contenant le graphe transformé
- SAN.m4x : le fichier contenant les structures et fonctions scicos liées au mécanisme de macro syndex.

A noter que le fichier SAN.sdx généré contient une architecture minimale avec un processeur (rootOperator) histoire de permettre la mise en place rapide de tests monoprocesseur.

D'autres fichiers sont générés afin de permettre de mettre rapidement en place des tests. Ces fichiers sont :

- SAN.m4
- SAN.m4m
- rootOperator.m4x
- GNUmakefile

Macro expansion et compilation :

Pour compiler (entendre par là : fichier M4 généré par syndex + fichiers générés pas le traducteur ⇒ [macroexpansion] ⇒ fichier C "macro étendu" ⇒ [compilation] ⇒ exécutable), il faudra fournir plusieurs fichiers :

- chaque fichier contenant la fonction (de type 4) correspondant aux blocs scicos doit être fourni dans le répertoire SAP/SCICOS_FILES,
- les fichiers C Scicos incluent plusieurs bibliothèques, il est nécessaire de les fournir dans SAP et, par soucis de compatibilité, également dans un répertoire SAP/scicos. Ces fichiers sont : machine.h et scicos_block.h (en théorie fournis avec l'archive contenant les fichiers C Scicos),
- le fichier s2s_common.c est un fichier (fourni avec le traducteur) contenant un code basique pour les capteurs et actionneurs ainsi que du code global à toute application scicos. Il doit également être inclu dans SAP/SCICOS_FILES

L'archive s2s_scicosFiles.tgz contient tous les fichiers décrits ci-dessus et doit être décompressée dans le répertoire SAP.

1.4 Procédure pour tests monoprocesseur

Le traducteur génère tout le code nécessaire pour amener la chaine de développement jusqu'à la compilation. Cependant certaines opérations sont pour l'instant à réaliser manuellement.

Noter que les points marqués *[étape supprimée]* correspondent à des points maintenant automatisés ou corrigés par la dernière version du traducteur. Ils sont laissés là au cas où un bug serait rencontré.

1.4.1 Procédure d'utilisation :

1. spécifier un schéma sous Scicos, créer un superbloc respectant les conditions de monoactivation/synchronisme puis cliquer sur Object ⇒ TO SynDEx,
2. *[étape supprimée]* éditer le fichier .sdx généré avec votre éditeur préféré. Couper la définition de l'opération clkRoot_<quelquechose> et la coller juste avant la ligne indiquant : main algorithm clkRoot.
3. lancer syndex sans oublier de lui donner le chemin des librairies (option : -libs libs/) puis ouvrir le fichier SAN.sdx. Sélectionner le processeur présent dans l'architecture, éditer sa définition, ouvrir ses "code generation phases" et rajouter la phase "init".
4. Lancer l'adéquation puis la génération de code
5. ne pas oublier d'ajouter les fichiers comme décrit dans la section précédente
6. si besoin est avant macro expansion :
   - personnalisez s2s_common.c (sensors et actuators),
   - changer le nom de la machine cible dans SAN.m4m (la votre),
   - *[étape supprimée]* changer les chemins pour des chemins corrects dans GNUmakefile,
7. faire "make rootOperator.c", notez que la macro expansion de rootOperator.m4 vers rootOperator.c peut se passer sans problème même si les fichiers C scicos ne sont pas là, c'est la compilation de rootOperator.c qui retournera alors une erreur,
8. *[étape supprimée]* editer rootOperator.c, chercher la ligne commençant par "dnldnl" et l'effacer.
9. Problèmes pouvant être rencontrés :

- en multiprocesseur, vous assurer avoir les droits sur la machine cible (sous linux, créer au besoin un fichier .rhost et le remplir de manière adéquate),
- suivant la version de SynDEx utilisée, la code m4 généré peut contenir des lignes commençant par "#", les supprimer le cas échéant

10. faire "make SAN.run" ce qui devrait lancer la procédure de compilation du fichier rootOperator.c et exécuter l'application.

1.4.2 Le fichier s2s_common.c

Le fichier s2s_common.c doit contenir les fonctions correspondant aux capteurs et actionneurs obtenus à la suite de la traduction d'un schéma Scicos ainsi que tout code commun externe pouvant être utilisé par un ou plusieurs des blocs présents dans l'application traduite.

Un code est fourni à titre expérimental afin que la chaîne allant de la conception du schéma Scicos jusqu'à la compilation et l'exécution du code généré par SynDEx puisse être testée rapidement. L'utilisateur est cependant encouragé à fournir le code correspondant aux capteurs et actionneurs qu'il souhaite utiliser.

1.5 Vers une application multiprocesseur

Le makefile, ainsi que d'autres fichiers générés par le traducteur Scicos/SynDEx sont prévus à des fins de test monoprocesseur (cf. manuel du traducteur), on ne pourra pas compiler une appli multiprocesseur à moins d'y faire les modifications nécessaires :

Soit votre application de nom : "mon_application", le traducteur a généré automatiquement 1 processeur appelé "rootOperator" ainsi que des fichiers nécessaire à la production du code C lui étant associé. Admettons que vous gardiez dans votre architecture ce processeur et souhaitiez ajouter 2 processeurs de noms "proc2" et "proc3" le tout relié par un bus TCP. Il faut alors :

- dans GNUmakefile modifier la règle "all" en rajoutant le fichier .c qui sera généré pour vos processeurs supplémentaires :
  all : $(A).mk rootOperator.c proc2.c proc3.c
  Notez que si vous souhaitez exécuter votre application, il vous suffit de faire : make mon_application.run
- créer un fichier proc2.m4x et proc3.m4x contenant :
  `include(mon_application.m4m)`
- modifier mon_application.m4m en rajoutant le nom réel des machines que vous souhaitez utiliser :
  ```
  define(‘rootOperator_hostname_’, nom_de_machine1)dnl
  define(‘proc2_hostname_’, nom_de_machine2)dnl
  define(‘proc3_hostname_’, nom_de_machine3)dnl
  ```
- aussi, notez qu'il est impératif que les machines distantes acceptent des connexions via rsh. Si ce n'est pas le cas, l'exécution bloquera au moment d'envoyer le code à exécuter sur les machines distantes.

Notez que cette méthode fonctionne aussi dans le cas d'une application non générée par le traducteur.

1.6 Derniers changements

21/03/05 :

- prise en compte des mémoires (dollar) dans le traducteur et résolution des problèmes de boucle algébriques,
- modification de l'interface. Ajout de la demande du chemin d'installation de SynDEx et automatisation de la création du GNUmakefile,
- le fichier "s2s_common.c" est maintenant inclu en début du fichier C et inclue tous les .H nécessaires aux fonctions Scicos. Les fonctions Scicos n'ont donc plus à inclure leurs propres .H (la fonction dmmul est également intégrée dans s2s_common.h ce qui allonge le fichier C résultant de manière conséquente).

08/04/05 :
- modification du fichier "s2s_common.c" pour qu'il permette la lecture/écriture dans un fichier. Cf. les commentaires du code pour l'utilisation.

10/06/05 : release Traducteur v2.0
- livraison de la version 2.0 du traducteur prenant totalement en compte le contrôle Scicos,
- génération du fichier .sdx seulement.

04/07/05 : release Traducteur v2.1
- le traducteur génère maintenant les macros nécessaires à la phase de macro expansion et compilation du code m4 généré par SynDEx, permettant de tester la chaine de développement dans son ensemble.

15/09/05 : release Traducteur v2.2 (beta)
- les blocs continus sont maintenant pris en compte par le traducteur lors de la génération du macro code m4 correspondant à l'exécution de ces blocs. Un solveur de type Euler est implanté et appelé lors de l'exécution de tels blocs,
- correction de bugs qui tendaient à provoquer des erreurs dans des schémas complexes,
- amélioration du code du système de prise en compte des paramètres des blocs Scicos dans SynDEx, le code est maintenant plus générique et permet facilement la prise en compte de paramètres supplémentaires si besoin est.

25/10/05 : release Traducteur v2.2.1
- la robustesse du traducteur concernant les blocs continus a été améliorée, l'utilisation d'un schéma mélangeant contrôle et blocs continus a été testé avec succès,
- la détection de blocs codés en Scilab a été élevée au rang de "error" plutôt que celui de "warning".

03/11/05 : release Traducteur v2.2.2
- le traducteur fonctionne maintenant avec la version de Scilab 3.1.1.

10/01/06 : release Traducteur v2.3
- le traducteur fonctionne maintenant avec la version de Scilab 4,
- ajout du paramétrage de la valeur du pas discret et la valeur du pas continu, du nombre d'itérations que doit exécuter l'application en temps réel, du chemin d'accès à la librairie de fonctions C Scicos,
- l'utilisation de fichiers (en lecture ou en écriture) par les opérations correspondant aux capteurs et actionneurs de SynDEx a été automatisée.