Programmation réactive probabiliste

Guillaume Baudart[1] et Christine Tasson[2]

[1]Université Paris Cité, INRIA, Paris, 75013, France
[2]ISAE-SUPAERO, Toulouse, France

Les langages de programmation synchrones ont été introduits pour la conception de systèmes embarqués. Aujourd'hui le langage industriel SCADE [Est19] est couramment utilisé pour la conception de systèmes embarqués comme les commandes de vol d'un avion ou le système de freinage d'un train. Lors de la conception de tels systèmes, il est primordial de prendre en compte l'incertitude due à l'environnement, aux capteurs, ou aux défauts matériels. Malheureusement, les langages synchrones existants n'offrent que peu de support pour modéliser des comportements probabilistes. Les langages de programmation probabilistes permettent de décrire des modèles probabilistes et proposent des méthodes automatiques pour inférer les paramètres du modèle à partir d'observations statistiques. ProbZelus est un langage qui combine programmation réactive synchrone et programmation probabiliste.

Dans la première partie de cet exposé nous présentons ProbZelus [BMA+20], une extension probabiliste du langage synchrone Zelus [BP13] (un langage académique proche de SCADE) qui permet de décrire des modèles probabilistes réactifs en interaction avec un environnement observable. Lors de l'exécution, un moteur d'inférence bayésien permet d'apprendre les distributions de paramètres du modèle à partir d'observations. Nous introduisons ProbZelus avec quelques exemples concrets avant de discuter la conception du langage. Nous détaillons ensuite les algorithmes d'inférence semi-symbolique au cœur de la machine d'exécution qui mêlent méthodes d'échantillonnage approximatives et calculs symboliques exacts.

Dans la seconde partie de cet exposé, nous présentons la sémantique formelle de ProbZelus. La sémantique des programmes synchrones peut être exprimée de différentes façons équivalentes. La sémantique co-itérative décrit des machines à états [CMPP23]. La sémantique relationnelle vérifie que les flots d'entrée et de sortie satisfont les contraintes du programme [BBD+17]. D'un autre côté, la sémantique probabiliste interprète les programmes à l'aide de distributions de probabilités [Koz81, Sta17]. Nous montrons comment combiner ces points de vue pour étendre les sémantiques synchrones au cadre probabiliste et ainsi raisonner sur l'équivalence de programmes. Nous pouvons alors démontrer la correction d'une passe de compilation nécessaire pour utiliser un algorithme d'inférence efficace pour les modèles mélangeant des paramètres d'espace (qui n'évoluent pas dans le temps) et d'états (qui varient).

Remerciements Ce travail a été financé par le projet émergence de la mairie de Paris, ReaLiSe. Merci à Louis Mandel et Thomas Ehrhard pour les nombreuses discussions fructueuses.

Références

[BBD+17] Timothy Bourke, Lélio Brun, Pierre-Évariste Dagand, Xavier Leroy, Marc Pouzet et Lionel Rieg : A formally verified compiler for lustre. *In PLDI*, 2017.

[BMA+20] Guillaume Baudart, Louis Mandel, Eric Atkinson, Benjamin Sherman, Marc Pouzet et Michael Carbin : Reactive probabilistic programming. *In PLDI*, 2020.

[BP13] Timothy Bourke et Marc Pouzet : Zélus : a synchronous language with ODEs. *In HSCC*, 2013.

[CMPP23] Jean-Louis Colaço, Michael Mendler, Baptiste Pauget et Marc Pouzet : A constructive state-based semantics and interpreter for a synchronous data-flow language with state machines. *In EMSOFT*, 2023.

[Est19] Esterel Technologies & ANSYS : SCADE Suite. `http://www.esterel-technologies.com/products/scade-suite/`, 2019.

[Koz81] Dexter Kozen : Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3) :328–350, 1981.

[Sta17] Sam Staton : Commutative semantics for probabilistic programming. *In ESOP*, 2017.