Raport tehnic şi ştiinţific

privind implementarea proiectului PN-III-P2-2.1-PED-2016-0494
Acronim: ForVer
Faza a doua: 1 ianuarie 2018 – 31 decembrie 2018

Răzvan Diaconescu

Implementarea fazei a doua a proiectului Formal Verification of Reconfigurable Systems a fost efectuată între 1 ianuarie 2018 şi 31 decembrie 2018. Rezultatele tehnice pentru 2018 sunt conforme cu contractul şi sunt:

- prototipul sistemului de specificare şi verificare pentru sisteme reconfigurabile, numit H,
- ghidul de utilizare pentru H,
- studiul de caz,
- pagina Web a sistemului.

Acestea sunt discutate în secţiuni separate, după cum urmează.

1 Sistemul de specificare şi verificare H

Sistemul H este implementat ca o extensie a Heterogeneous Tool Set (Hets), un pachet de tool-uri pentru parsarea, analiza statică şi managementul demonstraţiilor, folosit la specificarea si verificarea multi-formalism a sistemelor. Hets oferă suport pentru logica de ordinul întâi şi integrează demonstratoare automate de teoreme pentru logica de ordinul întâi.

Sistemul are mai multe componente, care pot fi descrise după cum urmează.

1. suport sintactic pentru declararea parametrilor procesului de hibridizare:
   - logica de bază,
   - simbolurile permise în cuantificări şi
   - constrângerile asupra clasei de modele pentru logica hibridizată.

Logica de bază este specificată folosind numele ei intern din Hets. Pe baza mecanismului de calificare din Hets, putem alege şi o sublogică a unei logici implementate în Hets, de exemplu logica algebrelor parţiale apare în Hets ca o sublogică a logicii CASL, şi o putem selecta cu notaţia `CASL.PFOL`. Clasa de morfisme folosite în cuantificare este determinată de tipul simbolurilor permise în lista de variabile. Acestea pot fi sau nominali sau tipuri specifice logicilor (de exemplu, constante sau constante rigide sau constante totale), date ca o listă de cuvinte. Pentru constrângeri am introdus o gramatică fixată, care acoperă toate tipurile de constrângeri care apar în exemple: asupra relaţiei de accesibilitate sau asupra interpretării simbolurilor în modelele locale.

Mai mult, definiţiile existente de logici hibridizate sunt înregistrate astfel încât noi constrângeri şi mai multe tipuri de simboluri permise în cuantificări pot fi adăugate într-un pas ulterior.

2. o metodă generică de a genera noi instanţe ale clasei de tipuri[1] **Logic** din Hets pe baza definiţiilor declarative de hibridizări de logici. Ideea principală este de a introduce tipuri polimorfice în Haskell pentru elementele unei logici hibridizate. De exemplu, signaturile unei logici hibridizate sunt compuse dintr-o signatură din logica de bază, o mulţime de nominali şi o mulţime de modalităţi cu arităţile lor. În Haskell le vom reprezenta astfel

```
data HSign sig = HSign {
                 baseSig :: sig,
                 noms :: Set.Set Id,
                 mods :: Map.Map Id Int}
```

unde `Id` este tipul de date folosit pentru identificatori[2]. Avantajul de a lucra generic este că ne putem referi, la signatura de bază dintr-o signatură a oricărei logicii hibridizate prin intermediul metodei `baseSig`. Pasul următor este de a da implementări generice pentru metodele din clasa de tipuri **Logic**. Aceste metode definesc funcţionalitatea logicii la mai multe niveluri: parsing, analiza statică, afişarea elementelor logicii în sistem, analiza sublogicilor unei logici etc. Pentru logicile hibridizate, implementările generice ale acestor metode sunt metode Haskell care au în lista de argumente şi rezultat tipurile polimorfice introduse la pasul anterior şi care au un context care determină condiţiile pe care trebuie sa le îndeplinească tipurile permise în instanţierea tipurilor polimorfice. Condiţia tipică este că ele trebuie să facă parte dintr-o instanţă a clasei de tipuri **Logic**, cu intuiţia că transmitem componentele din logica de bază a hibridizării. De exemplu, metoda generică pentru translatarea formulelor de-a lungul morfismelor de signaturi are tipul

[1] În Haskell, clasele de tipuri pot fi întelese ca interfeţele din Java. De exemplu, pentru a face un tip `t` o instanţă a unei clase de tipuri `Eq` pentru tipuri care admit egalitate pe baza unui operator `== : a -> a -> Bool` trebuie să definim o declaraţie de instanţă care introduce o metoda `== : t -> t -> Bool`.

[2] Mulţimea de modalităţi este reprezentată ca o funcţie pentru eficienţă.

```
mapSentence :: Logic lid sublogics basic_spec sen
                 symb_items symb_map_items sig
                  mor sym raw_sym proof_tree
            => lid ->
               GTypes.HMorphism sig mor ->
               GTypes.HFORMULA sen symb_items raw_sym ->
               Result (GTypes.HFORMULA sen symb_items raw_sym)
```

În timpul analizei definiţiei unei hibridizări pentru o anumită logică de bază, o nouă instanţă a clasei de tipuri **Logic** este generată pentru tipurile obţinute prin instanţierea variabilelor tipurilor polimorfice introduse la nivelul generic cu tipurile care apar în logica de bază. De exemplu, dacă `SignL` este tipul signaturilor pentru o logică L, tipul signaturilor într-o hibridizare a lui L este `HSign SignL`. Elementele noii instanţe a clasei de tipuri **Logic** sunt instanţe ale metodelor generice. Dacă tipurile pentru identificatorul logicii, signaturi, morfisme, formule, liste de simboluri şi simboluri necalificate pentru o logică L sunt respectiv `L, SignL, MorL, SenL, Symb_itemsL, Raw_SymL`, tipul funcţiei `mapSentence` va fi

```
L ->
GTypes.HMorphism SigL MorL ->
GTypes.HFORMULA SenL Symb_itemsL Raw_symL ->
Result (GTypes.HFORMULA SenL Symb_itemsL Raw_symL)
```

iar corpul funcţiei este moştenit de la implementarea generică. Procesul descris mai sus generează cod sursă Haskell, care este apoi compilat pentru a face noua logică hibridizată disponibilă pentru specificare în sistem.

3. suport pentru specificaţii structurate. Conform definiţiei limbajului de specificare H-spec, există o corespondenţă între specificaţiile structurate din H si cele din DOL, limbajul de structurare suportat în Hets. Pe baza acestei corespondenţe, este suficient să implementăm pentru logicile hibridizate acele metode ale clasei de tipuri **Logic** care sunt folosite în semantica structurării în DOL. Din nou, am realizat acest lucru în mod generic pentru construcţiile DOL folosite în H: redenumire de simboluri, uniuni de specificaţii, translatări de specificaţii de-a lungul morfismelor de signaturi, colimite de signaturi (necesare pentru translatarea propoziţiilor de-a lungul morfismelor de signaturi).

4. suport pentru metodologia de verificare. Pentru a obţine suport pentru demonstraţii pentru hibridizarea H a unei logici L, vom aplica procesul de ridicare a unei translatări din L în logica de ordinul întâi la o translatare din H în logica de ordinul întâi. Astfel, putem verifica dacă o conjectură în H este adevărată sau nu prin translatarea ei în logica de ordinul întâi şi verificarea ei cu ajutorul unuia din demonstratoarele automate de teoreme disponibile in Hets.

Am introdus o sintaxă declarativă pentru aplicarea acestui proces. Parametrii sunt mai simpli decât pentru hibridizarea logicilor şi nu necesită o sintaxă specială: numele translatării care este ridicată şi numele logicii hibridizate care va fi sursa translatării rezultate. Ultimul argument este necesar penttu că o logică admite mai multe hibridizări. Din nou, o metodă generică analizează aceste definiţii şi generează cod Haskell continând o noua instanţă a clasei de tipuri pentru translatări din Hets. Dupa o compilare a codului generat, noua translatare este disponibilă pentru demonstraţii prin translatare în logica de ordinul întâi în sistem.

5. suport pentru noi logici în Hets. Articolul [2] care dă fundamentele matematice folosite în dezvoltarea sistemului introduce şi un număr de exemple de logici hibridizate. Pentru a putea obţine in H suport pentru toate aceste logici, a fost necesar să implementăm în Hets a) logica algebrelor parţiale cu simboluri rigide b) logica de ordinul întâi cu simboluri rigide (ca o sublogică a primei logici), deoarece nu erau suportate în Hets iniţial.

Codul sursă al H este disponibil în mod liber pe o ramură de dezvoltare a Hets, la adresa `https://github.com/spechub/Hets/tree/rigid_casl`, şi adaugă mai mult de 7000 de linii de cod la implementarea Hets.

2 Ghidul de utilizare pentru H

Ghidul de utilizare al H este disponibil prin intermediul paginii Web a proiectului, la `http://imar.ro/~diacon/PN-III-P2-2.1-PED-2016-0494.html`. Ghidul începe cu instrucţiuni pentru descărcarea şi instalarea sistemului, apoi se introduce sintaxa limbajului de specificare: la nivelul logicilor de bază, se prezintă construcţiile sintactice pentru logicile care apar în exemplele din librăria H, logica propoziţională, de ordinul întâi, algebrelor parţiale şi variaţiile acestora cu simboluri rigide. Cele două notaţii ale limbajului de specificare H-spec, cea pentru matematicieni şi cea pentru dezvoltatori de sisteme, sunt apoi ilustrate, apoi se prezintă o listă de mesaje de eroare frecvente şi o listă de şabloane pentru proprietăţile tipice ale sistemelor reconfigurabile. Urmează prezentarea utilizării H pentru verificare. Se introduc convenţiile de sintaxă pentru dubla hibridizare, cazul special când logica de bază este o logică hibridizată. Construcţiile de limbaj pentru dezvoltarea modulară de specificaţii sunt prezentate cu ajutorul unor exemple, apoi se prezintă o funcţionalitate avansată a sistemului, adăugarea de noi logici hibridizate.

3 Studiul de caz

Am ales pentru studiul de caz specificarea unui sistem de control pentru un boiler cu aburi, un exemplu de dimensiuni medii care a fost folosit în competiţii [1] . Este important ca nivelul de apă din boiler să fie menţinut între o limită superioară şi una inferioară: în caz contrar, boilerul poate suferi defecţiuni majore. Acest lucru este asigurat

prin introducerea unor valori intermerdiare de control, la depăşirea cărora programul trebuie să acţioneze pentru revenirea la valori normale. Nivelul de apă poate fi controlat cu ajutorul a patru pompe de apă, fiecare prevăzută cu o unitate de control. Două componente de măsurare, una pentru nivelul apei şi una pentru cantitatea de aburi, sunt de asemenea prezente. Comunicarea între sistemul de control şi componente se face prin intermediul unui sistem de trasmisie a mesajelor, cu presupunerea ca transmisia are loc instantaneu. Dupa o fază de iniţializare în care se verifică funcţionalitatea componentelor şi se aduce nivelul de apă în boiler între limitele prevăzute, sistemul funcţionează în mod normal atâta timp cât toate componentele operează corect. In cazul unei defecţiuni a sistemului de transmisiune, a unităţii care măsoară cantitatea de aburi sau dacă nivelul de apă se apropie de limitele maximă sau minimă, sistemul efectuează o oprire de urgenţă. Dacă unitatea care măsoară nivelul de apă e defectă, sistemul continuă să funcţioneze folosindu-se de estimări ale nivelului de apă pe baza cantităţii de aburi şi a capacităţii pompelor. Dacă altă componentă e defectă, sistemul va continua să funcţioneze în modul de avarie până la remedierea defecţiunii.

Descrierea informală, în limbaj natural, a problemei conţine referinţe la diversele moduri de funcţionare ale sistemului şi cum diferite evenimente provoacă schimbări între aceste moduri, şi astfel sistemul apare în mod natural ca unul reconfigurabil.

Specificaţia noastră ilustrează toate caracteristicile limbajului de specificare introdus în proiect, atât la nivelul logicii hibridizate folosite pentru a specifica sistemul, o hibridizare a logicii de ordinul întâi cu simboluri rigide şi cuantificări pe nominali şi constante totale rigide, cât şi la nivelul specificaţiilor structurate şi a verificării. Proprietăţile sistemului pe care le putem verifica în H includ schimbarea de mod atunci când un eveniment are loc, dar şi că în toate stările sistemului într-un anumit mod de funcţionare proprietăctile de funcţionare asteptate au loc, de exemplu în modul normal de funcţionare toate componentele funcţionează corect. În formalizarea noastră, sistemul are 5 moduri si 9 evenimente, iar întreaga specificaţie are mai mult de 800 de linii. Studiul de caz este disponibil la `https://ontohub.org/forver/Sbcs.dol`.

4 Pagina Web a sistemului

Pagina Web a proiectului este disponibilă la adresa `http://imar.ro/~diacon/PN-III-P2-2.1-PED-2016-0494.html`. Prin intermediul acestei pagini, am facut disponibil H în două variante, câte una pentru fiecare notaţie. Versiunea curentă a prototipului oferă suport pentru logicile hibridizate introduse în [2]. Utilizatorii pot adăuga noi logici hibridizate folosind mecanismul declarativ descris anterior. Tot prin intermediul paginii proiectului se pot obţine manualul de utilizare, studiul de caz şi librăria de exemple.

References

[1] Jean-Raymond Abrial, Egon Börger, and Hans Langmaack, editors. *Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control (the*

Book Grow out of a Dagstuhl Seminar, June 1995)., London, UK, UK, 1996. Springer-Verlag.

[2] Răzvan Diaconescu and Alexandre Madeira. Encoding hybridized institutions into first-order logic. *Mathematical Structures in Computer Science*, 26(5):745–788, 2016.

Director de proiect,
Răzvan Diaconescu