MCS Project: PacMan

Bart.Bogaerts@cs.kuleuven.be
Ingmar.Dasseville@cs.kuleuven.be

22 oktober 2013

1 Inleiding

Het project gaat over het modelleren en verifiëren van een dynamisch systeem met behulp van IDP en NuSMV. Het bestaat uit 3 delen:

- In het eerste deel focus je op één toestand van het dynamisch systeem. Je stelt een (deel van het) vocabularium op en je formaliseert de wetten die in een toestand geldig zijn. Deze worden getest met het IDP systeem.
- In het eerste deel wordt de dynamische aspecten van het systeem gemodelleerd met behulp van het LTC-formalisme. Dit gebeurt door de oplossing van het eerste deel uit te breiden. Vervolgens voer je met IDP een aantal verificaties uit om de correctheid van je oplossing te testen.
- In het derde deel van het project stel je een model op voor hetzelfde dynamische systeem in de taal van NuSMV en je bewijst opnieuw een aantal invarianten.

In Sectie 2 beschrijven we het dynamische systeem. Daarna, in Sectie 3, beschrijven we de eerste opdracht.

2 Probleemstelling

Het onderwerp van dit project is PacMan. Een speelveld voor PacMan bestaat uit:

- Een aantal vakjes, waarvan sommigen met elkaar verbonden zijn.
- Per vakje een positie op een grid, gekenmerkt door een x en y coordinaat. Niet op elke positie staat een vakje.
- een aantal muren (tussen vakjes).
- een aantal goudstukjes (op vakjes).
- een geel happend ventje (PacMan).
- een aantal spookjes.

2.1 Vakjes, muren en richtingen

Vakjes worden aangeduid met een x- en y-coördinaat. De x-as loop horizontaal van links naar rechts. De y-as vertikaal van onder naar boven (een standaard assenstelsel dus). Er zijn vier mogelijke richtingen: Up, Down, Left en Right. Tussen twee vakjes kan zich een muur bevinden.

2.2 Bewegen

Zowel PacMan als spoken kunnen enkel van een vak naar een geconnecteerd vak lopen en kunnen niet van het bord lopen. Ze kunnen in vier richtingen bewegen. Één stap noemen we een move actie. Spoken en pacman doen op elke stap een move, tenzij dit onmogelijk is of tenzij het spel ten einde is. Ze kunnen in 1 stap hun richting niet omkeren: op een move volgt dus geen move in de tegengestelde richting. Ze kunnen enkel rechtdoor of haaks afslaan. PacMan kan wel zijn richting omkeren.

2.3 Goud en het einde van het spel

Op een aantal vakjes ligt initieel goud. Goud blijft liggen in een vak tot en met de eerste keer dat PacMan het vak betreedt. Het is dus mogelijk dat PacMan staat op een vakje met goud. Spoken eten geen goud. Het spel is gewonnen indien er geen goud meer ligt (alles dus is opgegeten door PacMan). Het spel is ten einde indien er geen goud meer ligt of totdat de positie van PacMan dezelfde is als die van een spookje. In het eerst geval wint de speler, in het tweede verliest hij. Indien zowel de conditie voor winnen als de conditie voor verliezen voldaan zijn, dan is het spel verloren. Zolang het spel niet ten einde is, moet iedereen op elk tijdstip bewegen.

3 Deel 1: modellering IDP

In het eerste deel van het project worden de wetten van een toestand gemodelleerd. Het is dus NIET de bedoeling om de evolutie van het dynamisch systeem te modelleren. Je gebruikt daarvoor het volgende vocabularium in IDP:

```
type dir
Down: dir
Left: dir
Right: dir
Up: dir

type xCo isa int
type yCo isa int

type agent
pacman:agent

NoPos(xCo,yCo)
Wall(xCo,yCo,dir)
```

```
/*Van wat hieronder staat kan een deel gegeven zijn, een deel gevraagd*/
Gold(xCo,yCo)
GameLost
GameWon
Position(agent,xCo,yCo)
PreviousMove(agent,dir)
Move(agent,dir)
```

Uiteraard mag je die vocabularium uitbreiden met nieuwe predicaten indien nodig! Je mag niets wijzigen aan de gegeven predicaten.
De betekenis van de symbolen volgt:

- `dir`: de vier richtingen. Voor de testen van uw specificatie zal de interpretatie van dir telkens bestaat uit alle en alleen de 4 richtingen.
- `agent` bestaat exact uit `pacman` en de spookjes.
- `NoPos(x,y)` drukt uit dat er op de plaats met coördinaten (x, y) geen vakje is (een diepe put als het ware).
- `Wall(x,y,z)` betekent dat in richting z van het vakje met positie (x, y) een muur staat. Bijvoorbeeld, `Wall(0,0,Up)` betekent dat er tussen vakje $(0, 0)$ en vakje $(0, 1)$ een muur staat.
- `Gold(x,y)` drukt uit dat het vakje op positie (x, y) goud bevat.
- `GameLost` en `GameWon` zijn waar als en slechts als het spel gewonnen respectievelijk verloren is.
- `Position(z,x,y)` drukt uit dat agent z op positie (x, y) staat.
- `PreviousMove(z,d)` drukt uit dat agent z vorige beurt in richting d bewogen heeft. (PreviousMove heb je enkel nodig om uit te drukken dat spookjes niet kunnen terugkeren)
- `Move(z,d)` drukt uit dat agent z deze beurt in richting d beweegt.

Een geldige toestand van het doolhof voldoet aan de volgende eisen:

- Alle vakjes zijn aaneengesloten: vanuit elk vakje bestaat er een pad dat niet door muren gaat naar elk ander vakje.
- `GameLost` en `GameWon` hebben de juiste interpretatie in functie van de andere predicaten.
- Iedereen is op exact 1 plaats.
- Niemand staat op plaatsen waar geen vakje is.
- Er kan enkel goud liggen op plaatsen waar een vakje is.
- Niemand beweegt door muren (bijvoorbeeld als PacMan de actie Move uitvoert naar boven, mag er boven hem geen muur staan).
- Iedereen kan hoogstens in 1 richting tegelijk bewegen.

- Afhankelijk van of het spel al gedaan is moet iedereen bewegen/mag niemand bewegen.
- Niemand beweegt van het bord.

Het is de bedoeling dat jij een theorie schrijft zodanig dat modelexpansie van jouw theorie met een (partiële) inputstructuur exact alle geldige toestanden oplevert. Je mag ervan uitgaan dat elke invoerstructuur aan de volgende eigenschappen voldoet:

- Alle types zijn geïnterpreteerd
- Het type "dir" is correct geinterpreteerd, net als de constanten Left, Right, Up en Down (je moet dus geen UNA[1] en DCA[2] constraints op dir uitdrukken).
- Het predicaat Wall is tweewaardig geinterpreteerd en bevat genoeg informatie om te weten waar er muren staan (je mag geen muren verzinnen). Het is wel zo dat dit predicaat niet per se volledig is. Bijvoorbeeld, een muur op $(0, 0, Up)$ zou betekenen dat er tussen vakje $(0, 0)$ en vakje $(0, 1)$ een muur staat. Impliciet staat er dus ook een muur **onder** vakje $(0, 1)$, maar dit is niet steeds gegeven. Indien je dit wil uitbreiden met de info dat er ook een muur onder $(0, 1)$ kan je zelf een eigen predicaat maken. **HINT:** doe dit!
- Het predicaat PreviousMove is gegeven en correct.

Uw oplossing is correct indien voor elke input met een structuur die minstens alle types specifieert, de correcte modellen berekend kunnen worden met modelexpansie. We zullen hiervoor enkele testvoorbeelden voorzien.
Merk op: indien de input niet vervolledigd kan worden tot een legale toestand (in plaats van een doolhof) van het PacMan spel, mogen er geen modellen zijn. Als het wel mogelijk is, dan moet modelexpansie evenveel modellen genereren als er mogelijke toestanden zijn.
Op Toledo vind je een skelet met onder andere een deel van het vocabularium voor de modellering. Het is natuurlijk de bedoeling dat je dit uitbreidt met andere benodigde relaties.
Op Toledo staan ook een aantal voorbeelddoolhoven. Aan de hand hiervan kan je je modellering testen, door bijvoorbeeld `idp pacman.idp instances.idp -e "check()"` te runnen, wat een lijst van geslaagde en gefaalde tests weergeeft.
Pro tip: Als je je variabelen steeds quantificeert, en typeert, dan zijn de warnings van IDP leesbaarder.

4 Praktisch

De output van dit deel van het project bestaat uit de volgende onderdelen:

[1] Unique Name Axioms: dit zou uitdrukken dat links en richts verschillende richtingen zijn
[2] Domain Closure Axioms: dit zou uitdrukken dat er geen andere richtingen bestaan dan diegene die gegeven zijn

1. Een beknopt verslag waarin je je designkeuzes toelicht; steek niet veel tijd in het verslag van deel 1, de modellering moet voor zichzelf spreken **en duidelijk van commentaar voorzien zijn**. Dit mag informeel zijn. Plaats in dit verslag aub hoeveel tijd je in dit deel van het project hebt gestoken.

2. Een bestand: "pacman.idp" dat je logische specificatie bevat.

3. Over je code:

    - Het systeem moet de verificaties correct uitvoeren voor de gegeven instances. Indien dit niet voor alle instances zo is, moet je duidelijk argumenteren waarom je modellering daar niet toe in staat is. Bij de quotering (van alle practica) worden ook extra verificaties uitgevoerd door ons, het is niet voldoende dat ze enkel werkt voor de training voorbeelden!
    - Het is verplicht om uit te gaan van de gegeven skelet-bestanden (en dus vocabularium, constraints en data over te nemen). Je moet de structuur van dit skelet respecteren.
    - Gebruik duidelijke namen voor geïntroduceerde symbolen en voor variabelen. (een naam x voor een y-coördinaat is vragen om problemen)
    - Schrijf commentaar: zorg ervoor dat wij elke regel idp code zonder problemen kunnen lezen.
    - Gebruik duidelijke indentatie.

Indien je vragen hebt, aarzel dan niet om op het discussieforum op Toledo iets te plaatsen of één van de assistenten te mailen.
Het project wordt in teams van twee gemaakt. Het resultaat wordt via Toledo ingediend, ten laatste 11 november om 23.59. Je oplossingen stuur je mee als zip-file met als naam "naam1_voornaam1.naam2_voornaam2.zip". Je moet dit slechts 1 keer indienen per groep.

Succes!