MCS Project: PacMan

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

21 november 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 tweede 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 Secties 3 en 4, beschrijven we de eerste opdracht. Sectie 2 is heel gelijkaardig aan de beschrijving in de opdracht van deel 1. De wijzigingen/verduidelijkingen ten opzichte van toen zijn aangegeven met "**Nieuw:**".

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 ten einde indien er geen goud meer ligt of indien 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. Nadien kan niemand nog bewegen.
Nieuw: Het is mogelijk dat er doodlopende gangen zijn in een opgave. In het geval dat een spookje zichzelf vastloopt in een doodlopende gang is Pacman gewonnen.
Nieuw: Indien een spookje en pacman elkaar kruisen (dit is mogelijk aangezien iedereen simultaan beweegt), is het spel ook ten einde. PacMan is dan verloren.

3 Deel 2: modellering IDP

3.1 Modellering

Het vocabularium voor dit deel is bijna identiek aan dat van het eerste deel; aan sommige predicaten is tijd toegevoegd en er zijn twee nieuwe predicaten:

- `I_gold(x,y)` is waar indien er initieel goud ligt op vakje x, y;
- `I_position(a,x,y)` is waar indien agent a initieel op positie x, y staat.

Je werkt best verder vanuit je modellering van deel 1. Hier zijn enkele hints hoe je die modellering nog kan verbeteren:

- Sommige studenten hadden bij deel 1 problemen met de reachability. Voor grote structuren duurde het soms lang om dit uit te rekenen. Je kan dit op twee manieren optimaliseren:

  - Schrijf geen regels van de vorm $\forall x, y : Reach(x, y) \leftarrow \exists z : Reach(x, z) \wedge Reach(z, y)$ maar gebruik $\forall x, y : Reach(x, y) \leftarrow \exists z : Edge(x, z) \wedge Reach(z, y)$.
  - Probeer niet om de volledige reachability uit te rekenen, maar leg 1 vakje vast en check dat alles bereikbaar is vanuit dat ene vakje.

- In deel 1 gaven we al de hint om een nieuw predicaat te maken dat uitdrukt wanneer ergens een muur staat (en dat volledige is dan de input). Je kan dit predicaat ook nog uitbreiden door overal aan de rand van het bord muren te verzinnen. Dit zal de rest van je modellering gemakkelijker maken; het feit dat je niet door muren kan lopen zal dan immers impliceren dat je ook niet van het bord kan lopen.

Het is de bedoeling dat je dit project maakt met het LTC formalisme: dus een fluent kan waar of vals "veroorzaakt" worden door het uitvoeren van acties en er is inertie op fluents. **Hint:** voeg deze "causes" predicaten toe indien nodig!

3.2 Verificatie

We vragen je ook om de volgende claims over je modellering na te gaan in de context van een gegeven structuur met eindige tijd.

1. het aantal muntstukken op het bord kan enkel omlaag gaan;
2. als het spel ten einde is, blijft het ten einde;
3. het is mogelijk om het spel te winnen;
4. het is mogelijk om het spel te verliezen.

Indien een van deze verificaties faalt, leg dan uit waarom.

3.3 Runnen en testen

Op ToLeDo vind je een file `doolhof-idp.zip`. Daarin zitten verschillende handige files, onder andere een skeleton waarvan je verder moet werken. Het skeleton vind je in `oplossing/solution.idp`. **Alles** wat je indient, moet in deze ene file staan. Alle andere idp-files zijn slechts om je te helpen bij het ontwikkelen, testen, etcetera. De andere files bevatten procedures voor het testen (`check()`) en visualiseren. Visualisatie werkt enkel onder Linux; deze files bevatten soms een pad naar een executable, die je, indien je niet in de pc klassen werkt, nog zelf juist moet zetten. De executables zelf kan je van op de pc klassen kopieren. Om de tests etcetera uit te voeren, doe je het volgende: `cd <projectdir>; cd vis; <idpexecutable> ../instances-and-testing/instances-and-testing.idp`, dus je voert idp uit vanuit de "vis" map. Idp zal dan zelf info printen over de beschikbare test- en visualisatie- procedures.
Het is de bedoeling dat je de verificaties uitvoert door de overeenkomstige theorie en procedure aan te vullen.
Belangrijk: plaats geen onnodige zaken in de ingediende file. Wij runnen immers automatische tests op jouw specificatie.

4 Deel 3: modellering in NuSMV

In deel 3 van de modellering is het de bedoeling dat je het dynamisch systeem modelleert in NuSMV.

4.1 Modellering

Een skelet van de verschillende modules is gegeven op ToLeDo. Het is de bedoeling dat jullie de modules main, agent en square aanvullen. Verander hierbij niets aan de gegeven variabelen. Je specificatie moet correct blijven als wij vakjes toevoegen of verwijderen!
Door de beperkingen van NuSMV moeten er een aantal dingen niet gemodelleerd worden:

- Je mag ervan uitgaan dat het gegeven doolhof volledig is: je moet niet controleren dat pacman niet van het bord valt (er zullen steeds muren aan de rand van het bord staan) en beide kanten van een muur zullen altijd gegeven zijn.
- Er moet niet gecontroleerd worden dat het doolhof samenhangend is.
- Doodlopende stukken mogen volledig genegeerd worden in jullie modellering (ga ervan uit dat deze niet bestaan).

4.2 Verificatie

Voer de volgende verificaties uit. Minstens 1 doe je met LTL en minstens 1 met CTL. Leg de verificaties altijd uit in de context van de doolhof zoals hij in het skelet gegeven staat.

- Vanuit elke begintoestand kan het spel zowel gewonnen als verloren worden.
- Muntjes kunnen niet terug verschijnen: als er ergens geen muntje ligt, dan zal in de toekomomst er nooit nog een muntje liggen. (check dit voor 1 concreet vakje)
- Het is mogelijk dat pacman oneindig vaak op locatie $(1, 0)$ is terwijl het spel nog bezig is.
- Als pacman oneindig vaak op locatie $(1, 0)$ komt en oneindig vaak op locatie $(1, 3)$ komt, dan wordt het spel nooit gewonnen.
- Eens het spel verloren of gewonnen is, blijft het spel in deze toestand.
- Er zijn minstens 10 tijdstappen nodig om het spel te winnen.

5 Praktisch

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

1. Een verslag waarin je je designkeuzes toelicht. Plaats in dit verslag aub hoeveel tijd je in dit deel van het project hebt gestoken.

2. Een bestand: “solution.idp” dat je IDP-specificatie bevat en een bestand “solution.smv” dat je NuSMV-specificatie bevat. **Respecteer deze bestandsnamen als je wil dat je automatische tests slagen**

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 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 12 januari 2014. 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!