Project MCS: Logical Pacman
Iteraties 2& 3

Jessa Bekker s0215494
Karel Moens s0215430

21 december 2013

1 Inleiding

In dit verslag worden kort de ontwerpbeslissingen bij het uitwerken van het tweede deel van het project toegelicht. De opdracht was ... Tot slot vermeldt de tekst ook de tijd die nodig was om dit te verwezenlijken.

2 IDP

2.1 Verbetering iteratie 1

Ondanks de correcte werking van onze oplossing voor iteratie 1, waren enkele verbeteringen op vlak van complexiteit en performantie mogelijk. Voor de performantie werd de voorwaarde dat alle vakjes elkaar moeten bereiken versoepeld naar dat 'e'en vakje alle andere moet kunnen bereiken, wat hetzelfde impliceert. Verder werd het predicaat `Edge(x1,y1,x2,y2,d)` toegevoegd dat verschillende voordelen geeft. Edge drukt de relatie uit tussen 2 aangrenzende vakjes op posities (x1,y1) en (x2,y2) waarvan vakje 2 in richting d ten opzichte van vakje 1 ligt. Dit wordt gebruikt om `Reach(x1,y1,x2,y2)` efficienter na te gaan door enkel naar buren te kijken en niet naar eender welk vakje. Het wordt ook gebruikt om mogelijke stappen te vinden (die niet door een muur, van het bord af of naar een NoPos gaan), wat gebruikt wordt als voorwaarde op `Move` en de 2 nieuwe regels: `DeadEnd` en `Crossed`. De implementatie wordt hierdoor minder complex.

2.2 GameWon en GameLost

`GameWon` en `GameLost` zijn niet als fluents geimplementeerd. De reden hiervoor is dat de natuurlijke wijze om regels over winnen/verliezen te verwoorden van het type zijn: "De huidige staat is ..., dus nu is het spel gewonnen/verloren". Bijvoorbeeld: "In de huidige staat zijn er geen muntstukken meer dus het spel is nu gewonnen." Gebruikmakende van een fluent zou de regel moeten verwoord worden als "In de huidige staat wordt een actie uitgevoerd waardoor het spel in de volgende staat gewonnen/verloren zal zijn", dus: "In de huidige staat is er nog maar 1 muntstuk, dat ligt op het vakje waar pacman op staat en pacman gaat een stap zetten dus het spel zal in de staat gewonnen zijn." Dit is duidelijk omslachtig. Er werd gekozen om de predicaten `GameWon` en `GameLost` de eenduidig te definiëren op de staat. Om dit te kunnen doen heeft de staat de predicaten `PreviousMove`, `PrevLost` en `PrevWon` nodig. `PreviousMove` is nodig om te bepalen of pacman en een geest elkaar net gekruisd zijn, `PrevWon` en `PrevLost` worden gebruikt om op te dragen dat eenmaal een spel verloren/gewonnen is, dit niet meer veranderd.

2.3 Verificatie

De eerste twee verificaties zijn voorwaarden waar altijd, in alle mogelijke modellen, aan voldaan moet zijn. De manier om dit na te gaan is contradictie: de omgekeerde voorwaarde toevoegen. Als de theorie met deze bijkomende voorwaarde satisfieerbaar is, betekend dat dat aan de originele voorwaarde niet voldaan werd. Als er zo geen modellen bestaan dan is het wel correct.

De laatste twee verificaties stellen dat er een model moet bestaan waarin aan een voorwaarde voldaan is. Om dit te controleren wordt de voorwaarde toegevoegd en nagegaan of er een model voor bestaat. Het verschil tussen de eerste twee verificaties en de laatste twee is dat het bij de eerste in alle modellen voldaan moet zijn en in de laatste in minimum 1 model.

De eerste verificatie (het aantal muntstukken op het bord kan enkel omlaag gaan) gebeurd door de voorwaarde toe te voegen dat er een op een tijdstip geen goud ligt op een vakje en op een later tijdstip wel. Dit werd verkozen boven de letterlijke vertaling van de opgave omdat het het strikter (muntstukken kunnen ook niet van vakje veranderen) en performanter is. Deze striktere voorwaarde impliceert wel dat het aantal muntstukken op het bord enkel omlaag kan gaan.

3 NuSMV

4 Tijdsbesteding

Jessa:	IDP	10u
	NuSMV	2u
Karel:	IDP	?
	NuSMV	?

5 Besluit