nuXmv: Exercises - Part B*

Patrick Trentin
patrick.trentin@unitn.it
http://disi.unitn.it/~trentin

Formal Methods Lab Class, May 20, 2016

Università degli Studi di Trento

---

*These slides are derived from those by Stefano Tonetta, Alberto Griggio, Silvia Tomasi, Thi Thieu Hoa Le, Alessandra Giordani, Patrick Trentin for FM lab 2005/15

Contents

Exercise: Odd/Even Counter [1/2]

Implement a 5-bit counter that alternates counting all odd numbers from 31 to 1 *(e.g. 31, 29, 27, ..., 3, 1)* and counting all even numbers from 30 to 0 *(e.g. 30, 28, 26, 2, 0)*. Use a variable "**out**" to represent the output of the counter. Use five Boolean variables "**b0**", "**b1**", "**b2**", "**b3**", "**b4**" to represent the bits of the counter, from the least-significative to the most-significative ones. Initially, all bits are set to TRUE. The transition relation is described as follows:

- "**b0**" changes value only when all other bits are FALSE
- "**b1**" changes value at each transition
- "**b2**" changes value only when "**b1**" is FALSE
- "**b3**" changes value only when both "**b1**" and "**b2**" are FALSE
- "**b4**" changes value only when "**b1**", "**b2**" and "**b3**" are all FALSE

Exercise: Odd/Even Counter [2/2]

Model the 5-bit counter, express the following properties, and check with nuXmv that all properties are verified.

- it is necessarily always the case that, if out is 1, then at the next step the value of the counter is 30
- it is necessarily always the case that if out = 31 then in 5 iterations out will evaluate to 21
- it is always the case that b1 changes value at each iteration
- it is always the case that, if b1, b2 and b3 are all FALSE, then the next value of b4 is !b4
- infinitely often out is 0
- if out=30 then eventually in the future out=20

Contents

Exercise: Overflow Counter [1/3]

Implement a 3-bit counter which counts the number of times an input boolean variable "**bin**" changes value from `FALSE` to TRUE. Use three boolean variables "**b0**", "**b1**", "**b2**" to represent the bits of the counter, from the least-significant to the most-significant one. Use an output variable "**out**" to represent the value of the counter. Use a variable "**overflow**", with values in the set {`NO`, `YES`}, to keep track of a counter overflow event. Use a variable "**obin**" to keep track of the previous value of the input variable "**bin**", and an output variable "**rise**" to express the fact that "**bin**" changed value from `FALSE` to TRUE in the current step. Use an input boolean variable "**reset**" to reset the value of "**b0**", "**b1**", "**b2**" and "**obin**" to their initial value. Initially, "**b0**", "**b1**", "**b2**", "**bin**" and "**obin**" should be set to `FALSE`, while "**overflow**" should evaluate '`NO`'.

Exercise: Overflow Counter [2/3]

Implement, using the assign-syntax, the following transitions:

- "**obin**" is set to `FALSE` if "**reset**" is TRUE, and to "**bin**" otherwise
- "**b0**" is set to `FALSE` if "**reset**" is TRUE, it is set to "**!b0**" if "**rise**" is TRUE, and keeps its value otherwise
- "**b1**" is set to `FALSE` if "**reset**" is TRUE, it is set to "**!b1**" if "**rise & b0**" is TRUE, and keeps its value otherwise
- "**b2**" is set to `FALSE` if "**reset**" is TRUE, it is set to "**!b2**" if "**rise & b0 & b1**" is TRUE, and keeps its value otherwise
- "**overflow**" is set to `'NO'` if "**reset**" is TRUE, it is set to `'YES'` if "**rise & b0 & b1 & b2**" is TRUE, and keeps its value otherwise

Manually verify that the simulation works as intended.

Exercise: Overflow Counter [3/3]

Express the following properties, and have nuXmv verify that all properties are `FALSE`.

- CTL: it is necessarily always the case that infinitely often the counter is 0
- CTL: it is necessarily always the case that eventually the counter is always different than 0
- CTL: it is necessarily always the case that , if "**overflow**" is 'YES' in a given state then it also holds that "**overflow**" is 'YES' until "**reset**"
- CTL: it is necessarily always the case that when "**b0**", "**b1**" and "**b2**" are TRUE then from the next state eventually the value of counter will go back to 0
- LTL: if "**rise**" is TRUE infinitely often, then "**overflow**" is 'YES' infinitely often as well
- **Bonus Point**: explain why the latter formula is verified if CTL is used instead of LTL.

Exercises Solutions

- will be uploaded on course website within a couple of days
- send me an email if you need help or you just want to propose your own solution for a review

- learning programming languages requires practice: try to come up with your own solutions first!