Ambiente para Avaliação Automática de Robôs Virtuais: uma Forma de Apoio à Aprendizagem de Robótica

Emanuel Oliveira, Itamar Moraes, Iverson Pereira, Sidney de Carvalho Nogueira, Taciana Pontual Falcão

DEINFO - Universidade Federal Rural de Pernambuco (UFRPE), Rua Dom Manoel de Medeiros, s/n, Dois Irmãos, 52171-900 - Recife/PE - Brasil

{emanuel.osilva, itamar.moraes, iverson.luis, sidney.nogueira, taciana.pontual}@ufrpe.br

Abstract. *Programming environments for virtual robots provide a visual interface for teaching robotics and computer science. Nowadays, the main way for evaluating the correctness of robots in such environments is observing the robot behaviour during simulations. This paper presents Robot Checker, an environment that is able to automatically evaluate and give feedback about the behaviour of programs written in ROBO, a programming language for virtual robots. The goal of the environment is to facilitate students' comprehension about the behaviour of developed programs. An associated visual language is also proposed to support the interaction between teachers and students within the environment.*

Resumo. *Ambientes de programação de robôs virtuais fornecem uma interface visual para o ensino de robótica e computação. A principal maneira de avaliar o funcionamento dos robôs virtuais nestes ambientes é através da observação do comportamento dos robôs durante as simulações. Este artigo propõe o ambiente Robot Checker que é capaz de avaliar de forma automática e prover feedback sobre o funcionamento de programas escritos na linguagem de programação para robôs chamada ROBO. O objetivo do ambiente é facilitar a compreensão dos alunos sobre o comportamento dos programas desenvolvidos. Uma linguagem visual associada é também proposta para auxiliar a interação de professores e alunos no ambiente.*

1. Introdução

Atualmente, cursos superiores na área de Computação sofrem com evasão de alunos causada, entre outros fatores, por: falta de motivação, dificuldade de assimilação de conceitos de lógica, dificuldade de desenvolvimento de raciocínio lógico-matemático [Hinterholz 2009]. Neste cenário, a robótica educacional vem se popularizando em escolas e universidades [Bezerra Neto et al. 2015] como uma alternativa construcionista e lúdica para o desenvolvimento do pensamento computacional e de conceitos básicos de programação, estimulando também a autonomia e a criatividade dos estudantes.

Ambientes de robótica educacional têm a vantagem de prover uma representação concreta, seja ela por meio do comportamento de um robô virtual ou físico, dos

comandos abstratos descritos pelas linguagens de programação, ajudando os estudantes a visualizarem os efeitos de comandos e estruturas lógicas e assim compreender melhor o seu funcionamento [Lessa et al. 2015]. Em particular, ambientes de simulação de robôs virtuais vêm se popularizando pois permitem a simulação passo a passo do programa do robô em um mapa virtual sem a necessidade de um robô físico. Entre os ambientes existentes, destacamos o ambiente RoboMind[1], que possui uma linguagem de programação educacional chamada ROBO, similar a uma linguagem de programação imperativa de propósito geral.

A principal forma atualmente disponível para verificação do funcionamento de robôs virtuais é a observação dos passos do robô durante a simulação. Entretanto, não existem ambientes gratuitos que possam analisar e fornecer feedback automático sobre o funcionamento de um robô virtual, isto é, se o robô se comporta da forma esperada quando executado em um conjunto de mapas pré-definidos. Recentemente, o ambiente proprietário RoboMind Academy[2] disponibilizou uma funcionalidade que permite que os professores visualizem se os programas enviados pelos alunos estão corretos. O ambiente verifica automaticamente se os programas submetidos solucionam os desafios de robótica cadastrados previamente. Como limitação, o ambiente não permite que os professores cadastrem novos mapas e desafios.

Este trabalho propõe o Robot Checker, um ambiente gratuito para verificação automática de robôs virtuais desenvolvidos com a linguagem ROBO, visando apoiar a aprendizagem de conceitos de programação e robótica. O ambiente permite cadastrar problemas de robótica e os mapas associados, bem como especificar o comportamento esperado para o robô em cada mapa. A função principal do ambiente é fornecer feedback automático ao programador-aprendiz sobre o funcionamento dos programas enviados, considerando os mapas cadastrados. Uma linguagem visual é proposta para facilitar a especificação do comportamento esperado em cada mapa. Internamente, o ambiente usa uma abordagem baseada na tecnologia de verificação de modelos (*model checking*), capaz de analisar de forma precisa o funcionamento dos programas.

A seção seguinte dá uma visão geral do ambiente RoboMind e da linguagem ROBO. Na Seção 3, apresentamos o protótipo da interface do ambiente para avaliação de robôs virtuais chamado Robot Checker e a linguagem visual associada. Ainda nesta seção, mostramos como a verificação de modelos é utilizada para automatizar a verificação dos robôs. Conclusões e trabalhos futuros são apresentados na Seção 4.

2. O Ambiente RoboMind

RoboMind é um ambiente lúdico que permite a programação e a simulação de robôs virtuais em mapas (bidimensionais) formados por um conjunto de paredes e objetos; e objetiva facilitar o aprendizado de lógica de programação e o desenvolvimento do pensamento computacional. Neste ambiente o aluno pode escrever programas na

[1] http://www.robomind.net/pt
[2] https://www.robomindacademy.com

linguagem educacional ROBO, simular o passo a passo do robô no mapa virtual e transferir o programa para um robô físico.

A linguagem ROBO foi desenvolvida para facilitar e tornar simples o contato de alunos com conceitos básicos da Computação. Comandos da linguagem permitem enviar ações ao robô, como: movimentar, procurar obstáculos e cores ao redor, mover itens e pintar. A Figura 1 mostra um exemplo do uso da linguagem ROBO para resolver um exercício em que o robô deve buscar no mapa um objeto circular com quatro pontas (denominado *beacon*). No lado esquerdo temos um programa na linguagem ROBO capaz de encontrar o *beacon* no mapa do lado direito da mesma figura. Este mapa corresponde ao estado inicial da simulação dentro do ambiente RoboMind.

```
repeatWhile(not(frontIsBeacon()))
{
    if(frontIsClear()) {
        forward(1)
    }else{
        backward(1)
        if( flipCoin() ){
            right()
        }else{
            left()
        }
    }
}
```

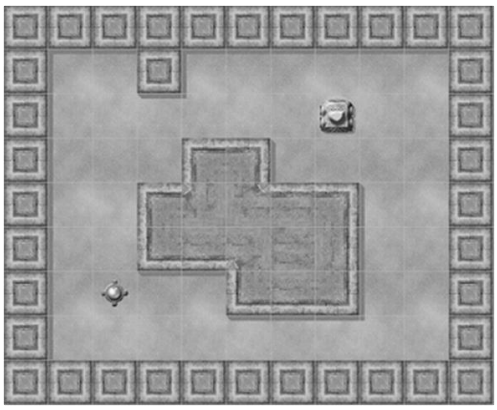

Figura 1. Exemplo de programa e mapa.

Os comandos *forward(n), backward(n), left() e right()* fazem com que o robô se movimente para frente e para trás (sendo *n* o parâmetro que informa a distância em número de células), e mude de direção para a esquerda ou direita, respectivamente. Em sua sintaxe, também há estruturas de repetição (*repeatWhile)* e condicionais (*if/else),* desse modo, permitindo controlar o fluxo do programa. A função *frontIsClear()* representa uma consulta ao sensor de obstáculos na frente do robô; retorna verdadeiro se não há nenhum obstáculo à frente. A função *flipCoin()* representa uma escolha aleatória entre os valores booleanos verdadeiro e falso, similar ao jogar uma moeda, permitindo realizar decisões não determinísticas no programa.

O programa descrito na Figura 1 movimenta o robô enquanto o *beacon* não é detectado à frente: se não há um obstáculo, o robô avança uma célula, caso contrário, o robô retorna uma célula e joga uma moeda para decidir se a orientação será para a direita (se o resultado for verdadeiro) ou para a esquerda (se for falso). Após observar a simulação na tela de RoboMind por vários passos seguidos, é possível ver que o programa alcança o *beacon*, sai do laço e termina.

3. Protótipo do Ambiente

Esta seção apresenta o protótipo de um ambiente online chamado Robot Checker, que automatiza a verificação dos robôs virtuais escritos na linguagem ROBO e é capaz de dar feedback automático para o aluno sobre os programas submetidos. O Robot Checker

é dividido em dois perfis de acesso (professor ou aluno). O perfil de professor tem a capacidade de criar turmas e propor exercícios. Em cada exercício, o professor inclui uma descrição do objetivo, o nível de dificuldade, uma lista de propriedades esperadas para os programas submetidos como solução, e os mapas a serem percorridos pelo robô. A interface permite que o professor acompanhe de forma remota as respostas dos alunos à medida em que forem enviando submissões.

A Figura 2 mostra uma tela do ambiente, contendo três colunas, que permite que os alunos editem e enviem soluções para um exercício. Todas as telas do protótipo estão disponíveis em http://bit.ly/2lzZpJ0. Na coluna da esquerda o aluno pode editar o programa a ser enviado como resposta para a atividade. O programa ilustrado na figura é o mesmo introduzido na Figura 1. Como alternativa, também existe um botão para importar o arquivo com a solução. Na coluna central, encontra-se uma imagem do mapa usado no exercício, e à direita, é mostrada a descrição do exercício. O mapa é anotado com elementos de uma linguagem visual (detalhada a seguir) que ajudam a especificar os comportamentos esperados pelo robô. O mapa na Figura 2 representa o mapa da Figura 1 com anotações da linguagem. Abaixo do mapa, é informada a lista de propriedades que devem ser satisfeitas para que o programa submetido esteja correto. Após enviar um programa, os itens da lista são marcados como corretos ou incorretos como forma de *feedback* dos erros e acertos realizados pelo programa submetido. Como exemplos de propriedades temos: verificar se o programa termina, ou seja, se está livre de loops infinitos, e se o robô conseguiu encontrar o *beacon* no mapa.

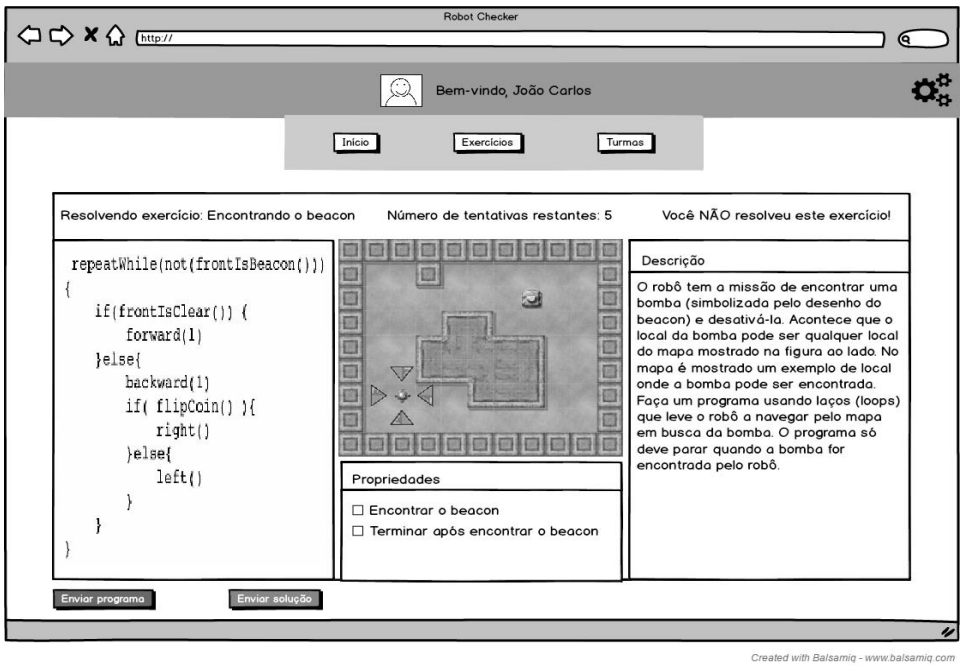

Figura 2. Tela no perfil do aluno para submissão de exercício.

3.1. Linguagem Visual

Um dos obstáculos para a verificação de robôs é que não existe uma forma padrão para descrever o comportamento esperado para o robô. Por exemplo, não existe um padrão

de como descrever o caminho a ser percorrido em um mapa, nem para descrever em qual ponto o programa do robô deve concluir sua sequência de comandos. Por estes motivos propomos uma linguagem visual para anotar o comportamento esperado do robô em um mapa. Os símbolos facilitam o entendimento dos alunos do que o professor espera como solução em cada exercício. A linguagem utiliza ícones e símbolos para indicar as posições do mapa onde o robô deve realizar ações. O ambiente Robot Checker processa automaticamente as anotações para verificar se uma solução submetida satisfaz o que é indicado pelos símbolos.

A Figura 3 ilustra os elementos da linguagem. No lado esquerdo da figura temos os principais símbolos organizados em duas linhas. Da esquerda para a direita, na primeira linha temos símbolos para indicar que: o robô deve andar uma única célula no sentido indicado; que o robô deve parar no sentido indicado; que o robô deve seguir na direção indicada enquanto não encontrar um obstáculo; e que o robô deve fazer um movimento de rotação de acordo com a direção da seta. Existem quatro variações para cada um dos símbolos da primeira linha, que correspondem às possíveis orientações do robô nas direções norte, sul, leste e oeste. Já na segunda linha, os símbolos indicam a posição inicial do *beacon* no mapa; o local onde o *beacon* deve ser capturado; o local onde o *beacon* deve ser liberado; e o ponto de parada do robô quando o programa concluir. No lado direito da Figura 3 é mostrada a ampliação do mapa exibido na Figura 2. As anotações nesta figura indicam que as soluções devem fazer o robô chegar até o *beacon* através de qualquer uma das quatro posições ao seu redor.

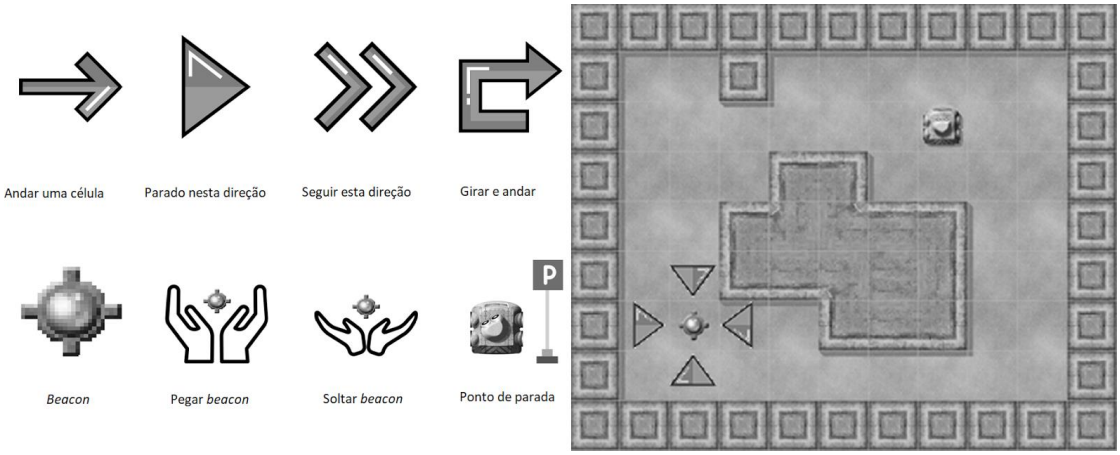

Figura 3. Símbolos da linguagem visual (esquerda) e exemplo de uso (direita).

O professor será responsável por adicionar os símbolos da linguagem visual no mapa: no momento em que ele insere um novo exercício para sua turma ele terá os símbolos a sua disposição, podendo adicioná-los com um simples arrastar e soltar. A ordem em que o robô deve executar cada uma das ações indicadas pelos símbolos será definida através de um número que será exibido acoplado ao símbolo. Por exemplo: para indicar que o robô precisa executar o movimento do Cavalo do xadrez, em formato de 'L', são necessários três movimentos seguindo a ordem: 1º para baixo, 2º para baixo e 3º para a direita. Neste exemplo, três setas seriam utilizadas para indicar as direções. A ordem de execução das setas é indicada pelos números 1 (para baixo), 2 (para baixo)

e 3 (para a direita). Quando nenhum número é indicado, como no exemplo da Figura 3, as ações podem ser executadas em qualquer ordem; neste caso, a primeira ação que acontecer é suficiente para concluir o exercício.

3.2. Verificação Automática

Após a submissão de um exercício, o ambiente utiliza a abordagem de verificação automática apresentada em [Nogueira et al., 2016] e resumida na Figura 4. As principais entradas para a abordagem são o texto do programa escrito em ROBO e o mapa no formato de RoboMind, contendo as anotações visuais. A primeira parte da abordagem consiste em obter a especificação formal (escrita em CSP) que representa o programa, o mapa e as propriedades a serem analisadas. CSP é o acrônimo para a notação formal chamada *Communicating Sequential Processes* [Roscoe 2011] criada para facilitar a especificação e análise da corretude de sistemas concorrentes e distribuídos. A especificação formal é obtida de forma automática, traduzindo cada elemento do programa e do mapa para o seu equivalente em CSP. A especificação CSP para o programa, o mapa e as propriedades esperadas são as entradas para uma ferramenta de verificação de modelos (*model checker*) chamada FDR [Gibson-Robinson et al. 2014]. Tal ferramenta analisa de forma exaustiva todos os caminhos e ações realizados pelo robô no mapa. A análise dá como resposta um veredito se o programa satisfaz (ou não) cada uma das propriedades. Esta resposta é apresentada na interface de Robot Checker indicando os itens da lista de propriedades (Figura 2) que são satisfeitos e quais não são.

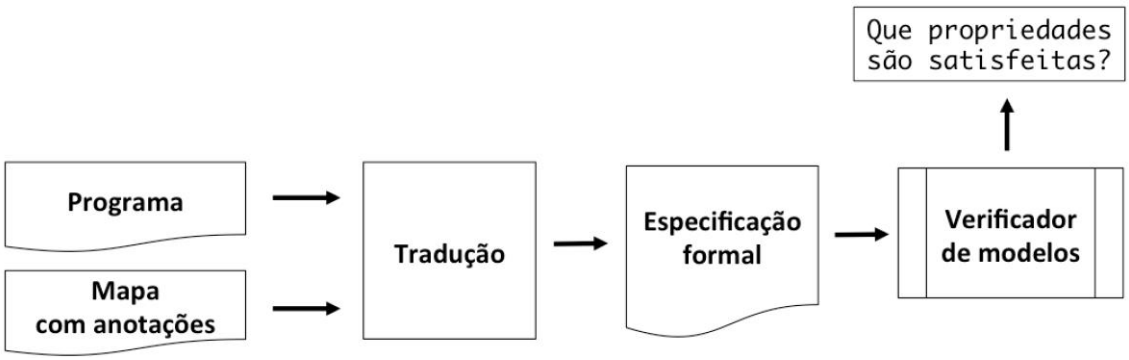

Figura 4. Visão geral da abordagem de verificação.

Para detalhes de como é feita a tradução para CSP, bem como a verificação, consultar [Nogueira et al. 2016]. As propriedades contempladas até o momento permitem verificar exercícios que tratam de: mover o *beacon*; fazer o robô seguir um caminho específico ou uma trilha colorida usando sensores de cor (evitando obstáculos); e achar o *beacon* [Nogueira et al. 2016].

4. Conclusões e Trabalhos Futuros

Este trabalho propõe um ambiente gratuito (Robot Checker) para a avaliação automática de robôs educacionais programados com a linguagem ROBO, cujo objetivo é auxiliar alunos de diversos níveis da educação, da básica à superior, no aprendizado de conceitos básicos de programação e robótica. Apresentamos um protótipo da interface de Robot

Checker e propomos uma linguagem visual utilizada pelo professor para descrever os comportamentos esperados para o robô em cada mapa. Internamente, o ambiente usa verificação de modelos para analisar os programas submetidos pelos alunos.

Sistemas de julgamento online (SJO) têm sido utilizados como base de dados para cadastrar desafios de programação e também como mecanismo de verificação automática (teste) de soluções candidatas, escritas em linguagens de programação de propósito geral, que são submetidas ao sistema. Na ocasião da escrita deste artigo, desconhecemos a existência de SJOs gratuitos para verificação automática de programas de robôs. Diferente dos SJOs, o ambiente proprietário RoboMind Academy só pode ser utilizado mediante pagamento, além disto, sua base de desafios fixa, isto é, não permite a inclusão de novos exercícios. Similar aos SJOs, o ambiente Robot Checker é gratuito, possui uma base de problemas que pode ser atualizada e é capaz de avaliar e dar feedback sobre as soluções submetidas de forma automática. Por outro lado, enquanto SJOs aceitam soluções escritas em linguagens de programação de propósito geral, e usam execução de testes para avaliar as soluções submetidas, Robot Checker aceita programas escritos em ROBO e usa verificação de modelos para analisar os programas.

Trabalhos futuros incluem a extensão do modelo para permitir a verificação de outras categorias de exercícios além das citadas na Seção 3.2, tais como programas que manipulam variáveis e problemas em que o robô deve pintar o chão. Além disso, planejamos implementar o ambiente de aprendizado para realizar experimentos controlados com professores e alunos, avaliando a contribuição educacional da proposta.

Referências

Bezerra Neto, R. P., Santana, A. M., Rocha, D. P., Souza, A. (2015) "Robótica na Educação: Uma Revisão Sistemática dos Últimos 10 Anos". Em *Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)*, Maceió - AL, Brasil.

Gibson-Robinson, T., Armstrong, P., Boulgakov, A., and Roscoe, A. W. (2014) "FDR3 - A Modern Refinement Checker for CSP". In: Tools and Algorithms for the Construction and Analysis of Systems, Edited by E. Ábrahám and K. Havelund, Lecture Notes in Computer Science, vol 8413, Springer, Germany.

Hinterholz, O. (2009) "Tepequem: uma nova ferramenta para o ensino de algoritmos nos cursos superiores em computação". Em *Anais do XVII Workshop sobre Educação em Informática (WEI 2009)*, Bento Gonçalves - RS, Brasil.

Lessa, V., Forigo, F., Teixeira, A. e Licks, G. P. (2015) "Programação de computadores e robótica educativa na escola: tendências evidenciadas nas produções do Workshop de Informática na Escola". Em *Anais do XXI Workshop de Informática na Escola (WIE 2015)*, Maceió - AL, Brasil.

Nogueira, S., Pontual Falcão, T., Mota, A., Oliveira, E., Moraes, I., and Pereira, I (2016) "An approach for verifying educational robots". In *Proceedings of Formal Methods: Foundations and Applications: 19th Brazilian Symposium* (SBMF 2016), Natal-RN, Brasil.

Roscoe, A. W. (2011). Understanding Concurrent System. Springer.