JOSÉ MARIA NOVAES DOS SANTOS

Dispositivos adaptativos cooperantes: formulação e aplicação

São Paulo
2014

JOSÉ MARIA NOVAES DOS SANTOS

Dispositivos adaptativos cooperantes: formulação e aplicação

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências

Orientador: Prof. Dr. João José Neto.

São Paulo
2014

JOSÉ MARIA NOVAES DOS SANTOS

Dispositivos adaptativos cooperantes: formulação e aplicação

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências

Área de concentração: Sistemas Digitais

Orientador: Prof. Dr. João José Neto.

São Paulo

2014

FICHA CATALOGRÁFICA

Santos, José Maria Novaes dos
Dispositivos adaptativos cooperantes: formulação e aplicação / J.M.N. dos Santos. -- São Paulo, 2014.
115 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sistemas híbridos 2.Aprendizagem computacional 3.Tecnologia adaptativa 4.Sistemas reativos I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

À minha esposa Juliana
e ao meu filho Matheus.

AGRADECIMENTOS

Agradeço à Deus pelo dom da vida, maior bem do ser humano. Aos meus pais pelos diversos ensinamentos, que hoje fazem parte do meu caráter e da minha moral.

À minha esposa Juliana, pelo apoio incondicional e compreensão durante todo o período de aulas, pesquisa e estudos. Ao meu filho Matheus, pelo carinho e amizade.

À minha família, pela torcida. Em especial meu irmão Renato, minha irmã Martha e minha cunhada Ana. Aos meus sobrinhos, David, Aline, Thalita, Larissa, Mariana e Thais pelo carinho. Ao meu irmão, um agradecimento especial pelos comentários e sugestões na primeira versão do artigo, o qual após as revisões foi aceito e publicado no congresso ICIEIS-2014.

Aos amigos Kiko, Milton e Jayme pela torcida e apoio. Ao meu afilhado Gabriel, pelo carinho e amizade. Ao amigo Danilo, agradeço o apoio e parceria nesse período.

À minha mãe, Maria Luiza, um agradecimento especial, pelas orações e torcida. Aos meus tios, pelo carinho.

Aos funcionários da universidade, em especial à Ana Maria pela revisão da bibliografia e formatação da tese.

Ao Prof. João, pela dedicação, compreensão e amizade. Foram inúmeras conversas, com o tradicional café e anotações em guardanapos, nas lanchonetes da Poli e da FEA.

Para finalizar, a todos os demais colegas, amigos e familiares, que direta ou indiretamente participaram dessa conquista.

Para Sempre

“Por que Deus permite
que as mães vão-se embora?
Mãe não tem limite,
é tempo sem hora,
luz que não apaga
quando sopra o vento
e chuva desaba,
veludo escondido
na pele enrugada,
água pura, ar puro,
puro pensamento.
Morrer acontece
com o que é breve e passa
sem deixar vestígio.
Mãe, na sua graça,
é eternidade.
Por que Deus se lembra
— mistério profundo —
de tirá-la um dia?
Fosse eu Rei do Mundo,
baixava uma lei:
Mãe não morre nunca,
mãe ficará sempre
junto de seu filho
e ele, velho embora,
será pequenino
feito grão de milho.”

Carlos Drummond de Andrade

RESUMO

Com a crescente complexidade das aplicações e sistemas computacionais, atualmente tem se tornado importante o uso de formalismos de várias naturezas na representação e modelagem de problemas complexos, como os sistemas reativos e concorrentes. Este trabalho apresenta uma contribuição na Tecnologia Adaptativa e uma nova técnica no desenvolvimento de uma aplicação para execução de alguns tipos de jogos, ("General Game Playing"), cuja característica está associada à capacidade de o sistema tomar conhecimento das regras do jogo apenas em tempo de execução. Com esse trabalho, amplia-se a classe de problemas que podem ser estudados e analisados sob a perspectiva da Tecnologia Adaptativa, através dos Dispositivos Adaptativos Cooperantes. A aplicação desenvolvida como exemplo nesta tese introduz uma nova ótica no desenvolvimento de aplicações para jogos gerais (GGP) e abre novos horizontes para a aplicação da Tecnologia Adaptativa, como a utilização das regras para extração de informação e inferência.

Palavras-chave: Sistemas reativos. Modelagem de sistemas reativos. Formalismos dirigidos por regras. Máquinas auto-modificáveis. Autômato Adaptativo. Dispositivos adaptativos cooperantes. Modelagem do comportamento de sistemas. Tabela de decisão. Jogos gerais. Dispositivos híbridos. Sistemas hibridos.

ABSTRACT

The complexity of computer applications has grown so much that several formalisms of different kinds became important nowadays. Many systems (e.g. reactive and concurrent ones) employ such formalisms to represent and model actual complex problems. This work contributes to the field of Adaptive Technology, and proposes a new approach for developing general game playing system, whose feature is the capability to play a game by acknowledging the game rules only at run time. This work expands the set of problems that can be studied and analyzed under the Adaptive Technology perspective, by means of cooperating adaptive devices. The developed application used a new approach for general game playing development bringing and widens the application field of Adaptive Technology with subjects related to information extraction and inference based in the devices rules.

Keywords: Reactive systems. Reactive systems modeling. Rule-driven formalisms. Self-modifying machines. Adaptive automata. Cooperating adaptive devices. Systems behavior modeling. Decision table. General Game Playing. Hybrid Devices. Hybrid systems.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da capacidade de processamento dos computadores modernos, armazenamento e velocidade dos computadores, os sistemas desenvolvidos nas mais diversas áreas de aplicação têm se tornado mais complexos.

Com a complexidade dos sistemas, a comunidade cientifica de engenharia de software tem se dedicado aos estudos envolvendo novas técnicas no desenvolvimento, gerenciamento e configuração desses sistemas. Um tópico que tem se tornado muito importante está relacionado com os sistemas autoconfiguráveis, que modificam seu comportamento e/ou estrutura em resposta à sua percepção do ambiente e de seus objetivos (LEMOS, 2013)

A adaptividade é um conceito relacionado à capacidade que tem um sistema de tomar a decisão de modificar seu próprio comportamento, em resposta ao seu histórico e estímulos de entrada, de forma autônoma (NETO, 2007).

Devido à forma como os sistemas adaptativos operam, eles se mostram particularmente adequados para a modelagem e representação de fenômenos de aprendizagem (NETO, 2011).

1.1 Justificativa

Com a criação de sistemas adaptativos, projetados para operarem de forma dinâmica, onde os requisitos, modelos e mudança de contexto mudam em tempo de execução, a área de verificação e validação de software têm se defrontado com grandes desafios (TAMURA, 2013).

Os sistemas adaptativos podem ser aplicados em sistemas de fluxos de trabalhos nas empresas, facilitando assim a frequente necessidade de mudanças pelas quais os processos de negócios empresariais passam. (BAEK; HAN; CHUNG, 2013).

Redes adaptativas têm sido utilizadas em ciências sociais, biologia, ecologia e ciências físicas (SAYAMA et al., 2013).

Os sistemas reativos podem ser considerados, sob uma perspectiva de abstração, como sendo “caixas pretas” que recebem entradas e em resposta fornecem saídas (INGÓLFSDÓTTIR et al., 2007).

Alguns trabalhos propõem a inserção da tecnologia adaptativa em sistemas reativos, como (NETO; ALMEIDA JR; SANTOS, 1998) e (NETO; ALMEIDA JR, 1999)

Os sistemas reativos estão presentes em várias áreas, como a engenharia de computação (INGÓLFSDÓTTIR et al., 2007), (THYSSEN; BENJAMIN, 2013) (VOGELSANG et al., 2012), a biologia (WOOLLEY; STANICKA; COTTER, 2013), (HOOD; GALAS, 2009), (ANTONY; BALLING; VLASSIS, 2012) e ciências sociais (SHANG, 2014).

Em alguns problemas temos várias soluções, cada qual com o seu tratamento específico, porém temos casos em que a aplicação pode ter uma solução melhor aplicando-se uma estratégia hibrida, combinando-se várias técnicas diferentes, ou para a mesma técnica, várias alternativas. Em linguagem natural, por exemplo, a verificação da classe gramatical de palavras pode ser feita por meio de um corpus de treinamento estatístico ou através de uma gramática. Uma alternativa é aplicação de um sistema hibrido, como o trabalho apresentado em (ROCHA, 2007).

Os robôs móveis têm sido utilizados em várias atividades atualmente, dentre elas auxilio em tarefas de escritórios, porém nas situações onde não há informações suficientes relativas ao ambiente, acrescentadas das incertezas decorrentes de situações reais, são necessárias funções de percepção e planejamento de alto nível. O trabalho apresentado em (LEE; CHO, 2014) ilustra o uso de soluções híbridas em robôs móveis, apresentando uma combinação de várias arquiteturas para gerar o comportamento do robô na camada de reação.

Devido à crescente necessidade da utilização de sistemas híbridos, tem sido desenvolvidos ambientes para modelagem, análise e controle para essa classe de sistemas, como o trabalho apresentado em (MAHEMUTI; YANG; LUO, 2014)

O processo de decisão está presente em várias áreas. O trabalho apresentado em (CHATTERJEE; CHAKRABORTY, 2014) apresenta o uso do processo de decisão na indústria de manufatura enquanto o trabalho (PILLAC et al., 2013) mostra a aplicação de tomada de decisão na área de transportes.

1.2 Objetivos

O objetivo desta Tese é apresentar uma formulação simples, rigorosa e inteligível, compatível com as técnicas clássicas envolvidas, em consonância com necessidades que ficam claras observando-se as dificuldades usuais na utilização de técnicas clássicas isoladas.

A aplicação desta técnica permite distribuir em vários dispositivos, cooperantes entre si, partes do fenômeno em estudo, possibilitando separar algum aspecto de interesse. Dessa forma, aplica-se nas regras do dispositivo de interesse uma análise do comportamento evidenciado, possibilitando o desenvolvimento de processos de aprendizagem e tomada de decisão.

1.3 Organização do Trabalho

Este trabalho está dividido em seis capítulos. O presente capítulo introduz a proposta da Tese, apresenta um panorama dos assuntos relacionados ao trabalho.

O capítulo 2 apresenta a formalização dos principais conceitos relacionados com esta pesquisa. O capítulo 3 apresenta a proposta conceitual e um exemplo ilustrativo para auxiliar a compreensão das definições apresentadas.

O capítulo 4 apresenta a descrição da aplicação desenvolvida, enquanto no capítulo 5 apresentam-se os resultados obtidos em experimentos práticos.

O capítulo 6 apresenta as contribuições e as conclusões desta pesquisa, finalizando com a indicação de uma série de propostas de trabalhos futuros.

2 CONCEITOS

Apresenta os principais assuntos relacionados a este trabalho, como a adaptatividade, sistemas para execução de jogos gerais (GGP), Statecharts e Tabela de decisão.

2.1 Adaptatividade

> “Adaptatividade é a capacidade que tem um sistema de, sem a interferência de qualquer agente externo, tomar a decisão de modificar seu próprio comportamento, em resposta ao seu histórico de operação e aos dados de entrada” (NETO, 2007)

O termo “dispositivo”, neste trabalho, refere-se a qualquer abstração formal cujo comportamento seja definido através de um conjunto de regras, que descreve a dinâmica de alteração da sua configuração.

Muitos formalismos tradicionais como, por exemplo, Máquina de estados finitos, Statecharts, Redes de Petri e Máquina de Moore, podem ser descritos através de conjuntos finitos de regras, que relacionam sua configuração atual com uma nova, em resposta a estímulos de entrada. Algumas configurações são escolhidas como configurações finais e indicam que as operações descritas nas regras que definem o dispositivo foram finalizadas com sucesso.

Se um dispositivo formal é adaptativo, seu conjunto de regras se modifica dinamicamente, sem ajuda externa, apenas em resposta a estímulos que ocorrem no ambiente e seu histórico de operação.

Em algumas situações, após a execução do modelo computacional representando várias situações do fenômeno estudado, conclui-se que os conjuntos que representam as regras e ações podem (ou devem) ser modificados para uma melhor adequação ou aperfeiçoamento do modelo computacional às suas condições correntes de operação. Nos dispositivos adaptativos, esta possibilidade

de modificação pode ser representada no conjunto de regras que definem o comportamento do dispositivo, através de regras adaptativas.

As regras adaptativas, quando executadas, alteram o conjunto de regras e ações, podendo desta maneira servir de suporte para a representação da experiência e dos fenômenos de aprendizado do dispositivo.

Em (ROCHA; NETO, 2001) os autores apresentaram uma prova conceitual de que os autômatos adaptativos possuem o mesmo poder computacional das máquinas de Turing.

Em 2003, Pistori apresentou em sua tese de doutorado (PISTORI, 2003) uma revisão e simplificação da notação, formalização e metodologia dos autômatos adaptativos.

Em 2005, foi apresentada a formulação de sistemas adaptativos de tomada de decisão (PEDRAZZI; TCHEMRA; ROCHA, 2005).

Em (PISTORI; NETO; PEREIRA, 2006), os autores apresentaram o formalismo de tomada de decisão, com mecanismo de inferência, denominado árvore de decisão adaptativa não-determinística.

Após a formalização inicial do autômato adaptativo, outros formalismos foram definidos mantendo as mesmas características, cujos comportamentos eram determinados por um conjunto de regras dinâmicas. A relação abaixo apresenta alguns exemplos desses dispositivos tais como:

- Autômato finito (PISTORI, 2003).
- Statechart (ALMEIDA JUNIOR, 1995).
- Tabelas de decisão (PEDRAZZI; TCHEMRA; ROCHA, 2005).

Maiores detalhes relacionados aos conceitos de adaptatividade podem ser encontrados em (NETO, 2001), (NETO, 2007), (CASTRO; NETO; PISTORI, 2007).

2.1.1 Dispositivos dirigidos por regras

Conforme a definição apresentada em (NETO, 2001), podemos representar um dispositivo guiado por regras (ND) como:

$$ND = (C, NR, S, c_0, A, NA) \quad \textbf{(1)}$$

onde:

- C representa o conjunto finito de todas as configurações possíveis que o dispositivo pode ter.
- c_0 é a sua configuração inicial ($c_o \in C$).
- A é um subconjunto de C, das configurações finais de aceitação ($A \subseteq C$).
- S é o conjunto de todos os estímulos possíveis de entrada, ($\epsilon \in S$).
- NA é o conjunto finito de todos os símbolos de saída, ($\epsilon \in NA$).
- NR é o conjunto de regras que descreve o comportamento do dispositivo ND, com $NR \subseteq C \times S \times C \times NA$. Uma regra r ($r \in NR$) tem a forma $r = (c_i, s, c_j, z)$, representando a mudança de estado c_i para c_j quando ocorrer o estímulo de entrada “s”, gerando o símbolo “z” como saída.

2.1.2 Dispositivos adaptativos dirigidos por regras

Um dispositivo adaptativo tem a capacidade de alterar de forma dinâmica suas regras. Segundo Neto (NETO, 2007) um dispositivo adaptativo pode ser decomposto em dois elementos, um dispositivo subjacente, tipicamente não adaptativo, e um mecanismo adaptativo, responsável pela incorporação da adaptatividade.

Assim, o comportamento do dispositivo adaptativo segue o comportamento do dispositivo subjacente equivalente, até a execução de alguma ação adaptativa não vazia, mudando a configuração do seu conjunto de regras. Associamos a

cada ocorrência das alterações do conjunto de regras a um contador inteiro k, iniciado com zero, e dessa forma o valor de k servirá para identificar assim cada passo onde ocorreu uma ação adaptativa.

Dessa forma, em cada passo k ($k \geq 0$) identificamos o dispositivo adaptativo (AD), conforme definição de Neto (NETO, 2001) por:

$$AD_k = (C_k, AR_k, S, c_k, A, NA, BA, AA) \quad \textbf{(2)}$$

onde:

- C_k representa o conjunto finito de todas as configurações possíveis em que o dispositivo pode se apresentar no passo k.
- c_0 é a sua configuração inicial ($c_o \in C$).
- A é o subconjunto de C, das configurações finais de aceitação ($A \subseteq C$).
- S é o conjunto de todos os estímulos possíveis de entrada, ($\epsilon \in S$).
- NA é o conjunto finito de todos os símbolos de saída, ($\epsilon \in NA$).
- AR_k é o conjunto de regras que descreve o comportamento do dispositivo AD no passo k, com $AR_k \subseteq BA \times NR \times AA$. Uma regra r ($r \in AR_k$) tem a forma r = (ba, c_i, s, c_j, z, aa), representando a execução da ação adaptativa anterior ba, a execução da regra da mesma forma que é executada no dispositivo subjacente, na forma (c_i, s, c_j, z), e finalizando, a execução da ação adaptativa posterior aa. As duas ações adaptativas, ba e aa pertencem respectivamente aos conjuntos BA e AA. As ações adaptativas, não vazias, alteram a configuração do conjunto de regras, através de inclusão e/ou exclusão de regras.

Em (NETO, 2001) podem ser encontrados os detalhes formais relacionados aos dispositivos adaptativos guiados por regras.

2.2 Tabela de Decisão

Uma tabela de decisão é um dispositivo guiado por regras, utilizado no auxilio de procedimentos voltados à tomada de decisão, com base em algumas condições.

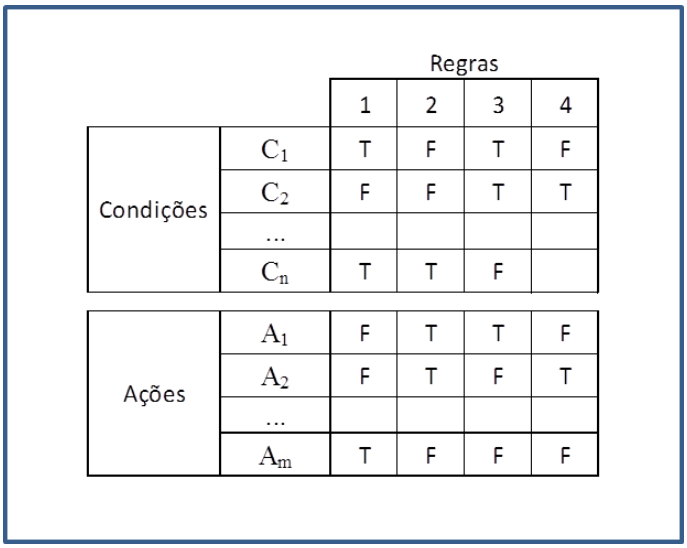
As tabelas de decisão podem ser usadas para facilitar a descrição de procedimentos que dependem de condições. Para seu uso, devem-se identificar as condições e ações envolvidas, elaborando-se as regras a serem executadas sempre que certas configurações forem exibidas.

A figura 2.1 representa uma tabela de decisão. Nesse exemplo, há quatro regras, numeradas de 1 a 4, na primeira linha quadriculada. Depois há um bloco indicando as condições, uma em cada linha. Se a condição deve ser satisfeita em uma regra, temos um “T” na célula relacionada à respectiva linha e coluna. Caso contrário há um “F”, indicando que a condição não deve ser satisfeita para a regra. Da mesma forma, o último bloco é constituído das ações que podem ser tomadas, uma em cada linha. Se a ação deve ser executada, a célula relacionada à respectiva linha e coluna é preenchida com “T”, caso contrário com “F”.

Uma regra da Tabela de decisão, na sua forma geral apresentada na figura 2.1, pode ter um número finito qualquer de condições que devem ser satisfeitas para a sua execução. Se as suas condições forem verdadeiras ela pode ter também um número finito qualquer de ações associadas à sua execução.

A figura 2.2 apresenta uma forma simplificada da representação da tabela de decisão. Nessa forma simplificada, as regras são constituídas de uma condição relacionada á configuração e outra ao estímulo de entrada, enquanto que temos apenas uma ação associada, representando a mudança de configuração. Nesse trabalho, usaremos a versão simplificada da tabela de decisão em alguns exemplos.

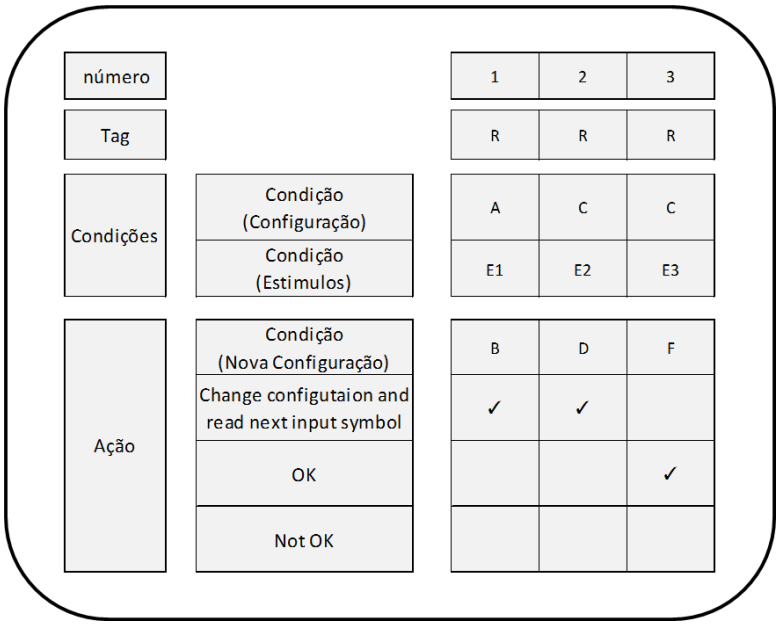
Figura 2.1- Estrutura de tabela de decisão



		Regras			
		1	2	3	4
Condições	C_1	T	F	T	F
	C_2	F	F	T	T
	...				
	C_n	T	T	F	
Ações	A_1	F	T	T	F
	A_2	F	T	F	T
	...				
	A_m	T	F	F	F

Fonte: (NETO, 2001)

Figura 2.2- Estrutura simplificada de tabela de decisão



número		1	2	3
Tag		R	R	R
Condições	Condição (Configuração)	A	C	C
	Condição (Estimulos)	E1	E2	E3
Ação	Condição (Nova Configuração)	B	D	F
	Change configutaion and read next input symbol	✓	✓	
	OK			✓
	Not OK			

Fonte: (NETO, 2001)

Para mais informações relacionadas à Tabela de Decisão Adaptativa, recomenda-se a leitura dos trabalhos (NETO, 2001), (OKADA, 2013), (PEDRAZZI; TCHEMRA; ROCHA, 2005) e (TCHEMRA, 2009).

2.3 Jogos GGP

A Inteligência Artificial é uma área de pesquisa que tenta simular os vários aspectos da inteligência humana, sendo a independência uma delas, abrangendo assim aspectos relacionados à tomadas de decisão de forma autônoma (OTTE, 2013).

“General Game Playing” (GGP) é uma sub-área da Inteligência Artificial cujo objetivo é a aplicação de técnicas de aprendizagem na criação de sistemas que tenham a capacidade de jogarem jogos apenas conhecendo-se as suas regras em tempo de execução, descritas em uma linguagem chamada GDL (“General Definition Language”).

Neste trabalho, sistemas com essas características serão denominados sistemas de jogos GGP.

Um agente de jogos GGP deve ser capaz de ler, como entrada, uma descrição de um jogo efetuado através de regras e conseguir jogar de forma autônoma (KUHLMANN; STONE, 2006).

Desde 2005, o grupo de Lógica de Stanford organiza anualmente uma competição de sistemas de jogos GGP, incentivando estudos relacionados à Inteligência Artificial.

Os estudos em jogos GGP desenvolvem tecnologias que podem ser aplicadas em outras áreas, como gerenciamento de processos de negócio, comércio eletrônico e operações militares (GENESERETH; LOVE; PELL, 2005).

A área de jogos GGP tem como objetivo projetar sistemas capazes de entender as regras de um novo jogo e usar tais descrições para jogar esses jogos efetivamente (CEREXHE; SABUNCU; THIELSCHER, 2013).

Os jogos GGP são síncronos e finitos. Por definição, todo jogo GGP tem ao menos um estado final, que pode ser atingido depois de um número finito de rodadas.

O desafio dos sistemas GGP é desenvolver um método de aprendizado de estratégias considerando apenas as regras. Em um jogo simples, há mais de mil estados possíveis distintos, o que torna pouco prática a tentativa de mapeamento de todas as possíveis configurações até um estado final.

Devido a enorme quantidade de combinações possíveis de lances, é difícil encontrar os estados da configuração que levam a um estado final. Assim, um dos grandes desafios é desenvolver métodos de aprendizagem de estratégias que se baseiam apenas nas definições das regras, que só são conhecidas durante a execução do sistema.

Assim, duas principais questões devem ser consideradas na aplicação GGP, a busca e a avaliação. A busca significa a capacidade em descobrir quais as possíveis configurações futuras, e avaliar cada uma delas. Assim, um grande desafio é considerar quais informações são relevantes na avaliação e devem ser considerados na construção.

2.4 GDL

Os jogos GGP são descritos através de uma linguagem chamada GDL (do inglês, “General Definition Language”).

A linguagem GDL é constituída das seguintes informações:

- **Regras de Configuração**

  São regras que definem a configuração do dispositivo. Em um jogo de damas, por exemplo, essas regras definem o tamanho do tabuleiro e que

em cada casa do tabuleiro pode conter uma peça branca, uma peça preta ou estar vazio.

- **Regras da entrada**

  Representa as entradas válidas para o dispositivo. No caso do jogo de damas, as entradas válidas são os lances. Podemos colocar qualquer peça, a branca ou a preta, em qualquer casa definida na configuração, cuja posição seja em uma linha posterior à posição atual, em relação às linhas das posições iniciais do jogo.

- **Informações iniciais**

  Essas informações definem as informações iniciais da configuração. No jogo de damas, indica as posições iniciais das peças pretas, das brancas e as casas em que ficam sem peças.

- **Regras de mudança de configuração**

  Essas regras determinam, com base nas configurações e entradas correntes, quais as próximas possíveis configurações válidas.

- **Definição de conceitos**

  São regras que definem conceitos que podem ser utilizados na condição de término de jogo, regra de pontuação e regras de validação. Por exemplo, no jogo da velha definimos o conceito de linha, coluna e diagonal, que são utilizadas nas condições de término e regra de pontuação.

- **Regras de validação**

  São regras que determinam quais entradas são válidas, com base na configuração atual do dispositivo. Por exemplo, no jogo de damas não se pode movimentar uma peça branca para uma posição onde há outra peça branca, embora a simples movimentação seja uma entrada válida.

- **Regra de pontuação**

  São regras que definem, com base em algumas condições, a pontuação, indicando se um objetivo foi atingido ou não. No caso de um jogo, indica quem foi o vencedor do jogo, ou se houve empate.

- **Condições de término**

  São regras que definem as condições de término do jogo. No caso do jogo de damas, por exemplo, se não há nenhuma posição do tabuleiro preenchida com uma peça branca, o jogo atingiu uma configuração final.

No anexo “A”, é apresentada a descrição, na linguagem GDL, do jogo da velha, para ilustrar os conceitos apresentados.

Para informações gerais sobre sistemas de jogos GGP e GDL, recomendam-se o trabalho (GENESERETH; LOVE; PELL, 2005) e (GENESERETH; THIELSCHER, 2014).

3 DISPOSITIVOS ADAPTATIVOS COOPERANTES - FORMULAÇÂO

Apresenta um breve histórico da formulação dos dispositivos adaptativos dirigidos por regras, a formulação dos dispositivos adaptativos cooperantes e um exemplo ilustrativo da formulação apresentada. Os dispositivos dirigidos por regras podem ser utilizados na modelagem de problemas complexos em Inteligência Artificial, Linguagem Natural e outras especialidades da computação, em que haja a necessidade de evidenciar uma alteração de comportamento em função de mudanças no ambiente com o qual o sistema está interagindo.

Em algumas situações a modelagem pode tornar-se mais compreensível quando efetuada com o auxílio de vários tipos de dispositivos, um grupo heterogêneo de dispositivos em que o comportamento de um pode afetar o comportamento dos demais.

A formulação apresentada nesse capítulo preenche a lacuna para representação formal de dispositivos adaptativos cooperantes, dirigidos por regras, nos quais o comportamento de um representante do grupo pode afetar o comportamento de outro dispositivo, resultando na alteração de suas regras. O objetivo é descrever formalmente os dispositivos e as interações entre seus componentes, mostrando a influência desse recurso sobre a sua expressividade global. Dessa forma, resulta uma formulação apropriada para descrever formalmente a representação e modelagem da influência de fatores externos sobre um dispositivo que pode reagir a eles através de mudanças comportamentais.

3.1 Histórico

A adaptatividade é, em um dispositivo formal, a característica ou recurso que lhe permite modificar seu comportamento de forma dinâmica, sem ajuda externa, apenas em resposta a eventos que ocorrem no ambiente, e em função do seu histórico de operação (NETO, 2007).

Na construção de dispositivos formais adaptativos dirigidos por regras, conceitualmente é possível explicitar duas componentes importantes: um dispositivo subjacente e um mecanismo adaptativo, responsável pela incorporação do recurso da adaptatividade (NETO, 2007).

Os dispositivos adaptativos dirigidos por regras foram formalmente definidos em (NETO, 2001), em que foi inspirado o resumo a seguir.

Define-se, em tempo de construção, um contador de tempo T com valor inicial zero e que é automaticamente incrementado por uma unidade cada vez que uma ação adaptativa não vazia é executada. Assim cada nome de um conjunto que varia no tempo é identificado pelo valor k assumido por T.

Dessa forma, pode-se dizer que um dispositivo AD_k= (ND_k, AM) é adaptativo quando para todos os passos de operação k, AD_k segue o comportamento do dispositivo subjacente ND_k até que alguma ação adaptativa, não nula, do mecanismo adaptativo AM seja executada iniciando-se então o (k+1)-ésimo passo de operações, alterando o conjunto de regras de ND_k, que passa a ser então o conjunto de regras do dispositivo subjacente ND_{k+1}. Um único incremento acarreta a execução de uma ação adaptativa anterior e uma ação adaptativa posterior. Ambas as ações adaptativas podem ser constituídas de um conjunto de ações adaptativas elementares, ou seja, podem ocasionar várias inclusões e/ou exclusões de regras no dispositivo.

O dispositivo AD inicia a sua operação em $\mathbf{c_0}$, com uma forma inicial AD_0 = (C_0, AR_0, S_0, $\mathbf{c_0}$, A , NA, BA, AA) . Em um passo k ≥ 0, um estímulo de entrada sempre altera o dispositivo AD para a próxima configuração iniciando a operação em (k+1) se e somente se alguma ação adaptativa não vazia foi executada. Então, em qualquer passo k (k ≥ 0) o dispositivo tem a forma:

$$AD_k = (C_k, AR_k, S_k, \mathbf{c_0}, A_k , NA, BA, AA). \quad (\mathbf{3})$$

Nessa formulação, temos:

- AD= (ND_0, AM) é algum dispositivo adaptativo, composto de um dispositivo subjacente inicial ND_0 e um mecanismo adaptativo AM,
- ND_k é um dispositivo não adaptativo de AD no passo k, sendo ND_0 o dispositivo subjacente inicial, definido pelo conjunto de regras NR_0 de regras não adaptativas. Por definição, qualquer regra não adaptativa em qualquer NR_k ($k \geq 0$) espelha a sua correspondente em AR_k,
- C_k é o conjunto de todas configurações possíveis para ND no passo k ($k \geq 0$),
- S é um conjunto finito de todos os possíveis estímulos de entrada considerados como eventos válidos de entrada para AD, sendo que o evento vazio pertence a S ($\epsilon \in S$),
- w é a cadeia de entrada de estímulos, e $w = w_1\, w_2\, w_3\, w_4 \ldots\ldots w_n$ ($w \in S$),
- c_0 pertence a C e é a configuração inicial do dispositivo ($c_0 \in C$),
- A_k é o conjunto das configurações finais (de aceitação), $A_k \subseteq C_k$,
- NA é o conjunto finito de símbolos de saída,
- AR_k é o conjunto de regras adaptativas, dado pela relação $AR_k \subset$ BA x C x S x C x NA x AA. As regras do conjunto AR_0 definem o comportamento inicial do dispositivo AD. Ações adaptativas mapeiam o conjunto corrente de regras adaptativas do dispositivo, AR_k, de AD, em um novo conjunto de regras AR_{k+1} adicionando e/ou excluindo regras adaptativas. As regras em AR_k são da forma ar = (ba, c_i, s, c_j, z, aa), significando que em resposta a algum estimulo $s \epsilon S$ ela executa inicialmente a ação adaptativa anterior, ba, em seguida a regra não adaptativa nr = (c_i, s, cj, z), e por fim a ação adaptativa posterior aa. Uma ação adaptativa pode eventualmente excluir a regra à qual está associada. Nessa situação, se a ação elementar de exclusão de regra pertencer à ação adaptativa anterior, a atividade de execução da regra será descontinuada assim que a ação adaptativa anterior tiver sido concluída.
- Define-se AR como o conjunto de todas as possíveis regras adaptativas para AD,

- Define-se NR como o conjunto de todas as possíveis regras não adaptativas subjacentes para AD,
- BA e AA são conjuntos de ações adaptativas, ambos podendo ser a ação adaptativa vazia ε (ε ∈ BA ∩ AA) e
- Define-se para um dispositivo AD particular o mecanismo adaptativo como o conjunto AM ⊆ BA x NR x AA, aplicado em qualquer passo k em cada regra em $NR_k \subseteq NR$. AM é de tal forma que opera como uma função aplicada a qualquer subdomínio $NR_k \subseteq NR$. Isso significa associar um par de ação adaptativa a cada regra não adaptativa. AR_k é o conjunto de todas as ações adaptativas que estão associadas às respectivas regras não adaptativas de NR_k.

3.2 Introdução

Os Dispositivos Adaptativos Cooperantes são constituídos de um grupo finito de dispositivos adaptativos dirigidos por regras, cujos dispositivos subjacentes podem ser de naturezas distintas, uma camada de comunicação e um mecanismo responsável pelo gerenciamento e coordenação das mensagens de comunicação entre os dispositivos. Cada dispositivo possui um módulo de comunicação e interpretação de mensagens, as quais podem ser trocadas entre qualquer par de dispositivos.

Cada dispositivo é composto de:

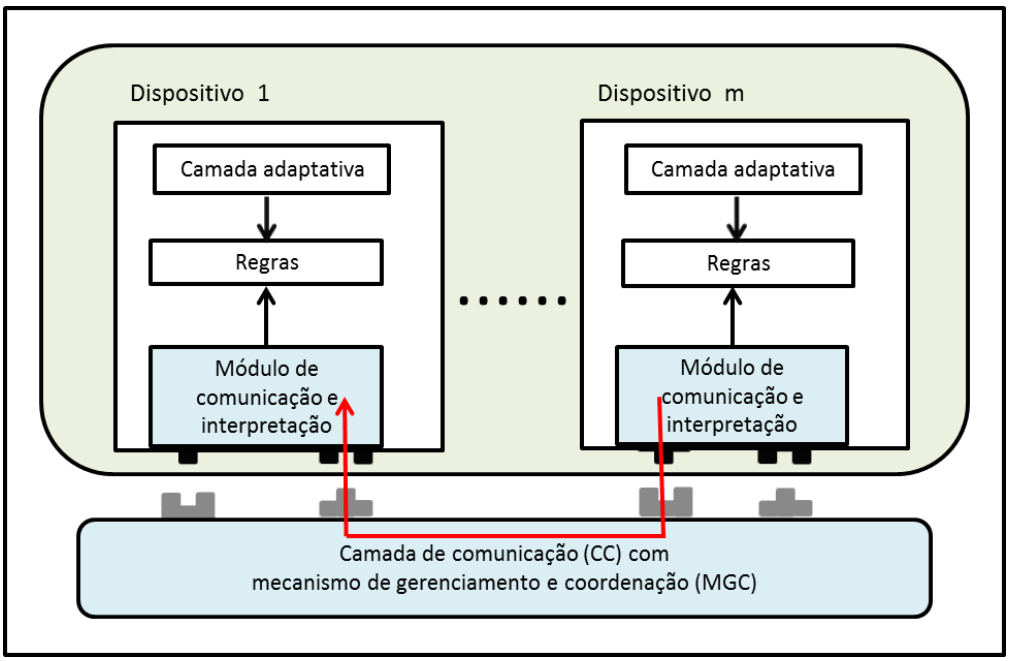
- Dispositivo subjacente dirigido por regras,
- Camada adaptativa e
- Módulo de comunicação e interpretação, capaz de enviar e receber mensagens de comunicação padronizadas.

O conceito de dispositivos adaptativos cooperantes pode ser entendido como sendo a propriedade que um dispositivo, pertencente ao um grupo finito de dispositivo, tem de alterar de forma autônoma o comportamento de outro dispositivo, pertencente ao mesmo grupo, baseado no seu comportamento ou em função de estímulos de entrada.

As mensagens de comunicação são gerenciadas e controladas por meio de um mecanismo de controle. A figura 3.1 ilustra o funcionamento da interação entre dois dispositivos. O dispositivo “m” envia uma mensagem ao dispositivo “1”, através da camada de comunicação, representada pela seta vermelha.

Tecnicamente podemos dizer que é o próprio dispositivo quem altera as suas regras, entretanto os dispositivos pertencente a um conjunto de dispositivos com a camada de comunicação possuem a capacidade de enviar mensagens a um outro dispositivo qualquer, do mesmo grupo considerado, solicitando-lhe que efetue alteração no seu conjunto de regras. Sob essa ótica, todo dispositivo do conjunto possui a capacidade de enviar e receber solicitação de alteração de regras. A comunicação entre os dispositivos é efetuada através de mensagens padrão, que compõem um protocolo de comunicação *PC*.

Figura 3.1- Comunicação entre dois dispositivos.



Fonte: Autor

3.3 Definição

Seja AD o dispositivo adaptativo definido por Neto em (NETO, 2001). Dispositivos Adaptativos Cooperantes podem ser representados por m (m>1) dispositivos adaptativos $\{AD^r\}$ para r=1,.., m, uma camada de comunicação (CC) e um mecanismo responsável pela coordenação e gerenciamento das mensagens de comunicação (MGC):

$$DAC = (\{AD^r \mid r=1,..,m\}, CC, MGC). \quad (\mathbf{4})$$

Ao grupo de dispositivos adaptativos AD^r (r=1,..,m) podemos denominar de G_mAD, e dessa forma:

$$DAC = (G_mAD , CC, MGC) \quad (\mathbf{5})$$

Onde:

$$G_mAD = \{AD^1, AD^2, ... ,AD^m\}. \quad (\mathbf{6})$$

Da mesma forma que aplicado ao comportamento dos dispositivos adaptativos, define-se para o dispositivos adaptativos cooperantes, em tempo de construção, um sequenciador T2 com valor inicial zero e que é automaticamente incrementado de uma unidade quando uma mensagem não vazia entre dois dispositivos é interpretada pelo módulo de comunicação e interpretação. Assim, cada nome de um dispositivo adaptativo dinamicamente modificável é identificado através de um diferente valor tk para a variável T2. Portanto, os dispositivos adaptativos cooperantes podem ser descritos como:

$$DAC_{tk}= (G_mAD_{tk}, CC, MGC) \quad (\mathbf{7})$$

ou,

$$DAC_{tk}=(\{AD^r_{kr,tk} \mid r=1,..,m;\ kr \geq 0 \text{ e } tk \geq 0\}, CC, MGC). \quad (\mathbf{8})$$

DAC_{tk} é dito cooperante quando, para todos os passos de operação tk (tk ≥ 0), DAC_{tk} segue o comportamento de todos os dispositivos adaptativos do conjunto $GmAD_{tk}$, até que a geração de alguma mensagem não vazia do mecanismo CC, enviada por um dispositivo "b" (b∈ {1,...,m}), seja interpretada por um dispositivo destino "r" (r∈ {1,...,m}) iniciando-se um novo passo de operações no (k+1)-ésimo passo, alterando o conjunto de regras do dispositivo AD^r (1 ≤ r ≤ m, 1 ≤ b ≤ m e r ≠ b), que passa a ser então o novo conjunto de regras dos dispositivos $GmAD_{tk+1}$.

De forma análoga ao que ocorre com as ações adaptativas, um único passo de incremento pode resultar na execução de dois grupos de mensagens, um anterior e o outro posterior à execução de regras. Os dois grupos são constituídos de mensagens elementares (mensagens padrão), de cuja execução podem resultar inclusões e/ou exclusões de regras do dispositivo, e são executados respectivamente antes e depois da execução da regra à qual estão associados, e ao menos um dos grupos de mensagens não é vazio.

Os dispositivos adaptativos cooperantes DAC iniciam a sua operação com uma forma inicial DAC_0, onde cada dispositivo adaptativo AD^r (r=1...m, m>1) tem uma configuração inicial relativa à camada de comunicação, quando o valor tk da fórmula (8) é zero, sendo expresso por:

$$DAC_0=(\{AD^r_{kr,0} \mid r=1,..,m \text{ e } kr \geq 0\}, CC, MGC) \qquad (\mathbf{9})$$

Na configuração inicial, representada por DAC_0, nenhuma mensagem entre os dispositivos ocorreu. Porém, cada dispositivo individualmente pode ter sofrido alterações em suas regras, decorrentes de suas ações adaptativas, indicadas por kr (r=1,…,m). Em um passo tk (tk ≥ 0), um estímulo de entrada altera o conjunto de dispositivos AD para a próxima configuração, iniciando a operação no (k+1)-ésimo passo se e somente se alguma mensagem não vazia da camada de comunicação, for executada. Então, para qualquer passo tk (tk ≥ 0), DAC_{tk} corresponde à configuração associada ao passo tk de todos os seus componentes.

Em qualquer combinação de passos kr (k1, k2, ...km) e tk (tk >= 0), cada um dos m dispositivos, (AD^r) para 1 ≤ r ≤ m , tem a forma:

$$AD^r_{kr,tk} = (C^r_{kr,tk}, IAR^r_{kr,tk}, S^r, c^r_{kr,tk}, A^r , NA^r, BA^r, AA^r, IBA, IAA) \quad (\mathbf{10})$$

Portanto, $(AD^r)_{tk}$, para r=1,..., m, representa a configuração de cada dispositivo adaptativo, de DAC, no passo tk sendo $(AD^r)_0$ a sua configuração inicial, definida pelo conjunto de regras $IAR^r_{kr,0}$, para qualquer passo kr (para kr ≥ 0), relativo à execução de ação adaptativa. Por definição, qualquer regra que não tenha mensagem da camada de comunicação (regra de $IAR^r_{kr,tk}$) espelha a sua correspondente em AR^r_{kr}. Assim temos:

- G_mAD_{tk} ={ $AD^r_{kr,tk}$ | r=1,..,m; kr ≥ 0 e tk ≥ 0} representa a configuração dos dispositivos adaptativos no passo tk, sendo $AD^r_{0,0}$ a sua configuração inicial. Cada dispositivo $(AD^r)_{tk}$ é o espelho do dispositivo adaptativo AD^r no passo tk , cada um deles em seu passo kr (kr ≥ 0, para r=1...m), relativo à execução de ação adaptativa.

- $C^r_{kr,tk}$ é o conjunto de todas as configurações possíveis para $AD^r_{kr,tk}$ no passo tk e kr (tk ≥ 0 e kr ≥ 0 , para r=1...m).

- IBA e IAA (para r=1,…,m) são grupos de mensagens entre dispositivos, ambos podendo ser vazios .

- S^r (para r=1,…,m) é um conjunto finito de todos os possíveis estímulos considerados como válidos de entrada para AD^r, sendo que o estímulo vazio pertence a S^r ($\epsilon \in S^r$); w^r é a cadeia de entrada de estímulos:

  $$w^r = w_1^r\, w_2^r\, w_3^r\, w_4^r \ldots\ldots w_n^r \quad (\mathbf{11})$$

  Onde $w_n^r \in S^r$, para r=1,...,m e n ≥ 0.

- $c^r_{0,0}$ pertence a $C^r_{0,0}$ e é a configuração inicial dos dispositivos DAC.

- A^r é o conjunto das configurações finais (de aceitação) dos dispositivos

DAC, $A^r \subseteq C^r$ (para r=1,...,m).

- NA^r é o conjunto finito de símbolos de saída dos dispositivos r (para r=1,...,m)

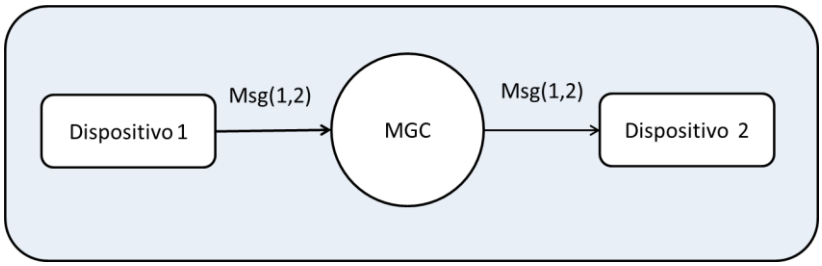
- $IAR^r_{kr,tk}$ é o conjunto de regras, dado pela relação $IAR^r_{kr,tk} \subseteq IBA^r \times BA^r \times C^r \times S^r \times Cr \times NA^r \times AA^r \times IAA^r$. As regras do conjunto $IAR^r_{0,0}$ (para r=1,...,m) definem o comportamento inicial do conjunto dos dispositivos DAC. As mensagens entre os dispositivos alteram o conjunto corrente de regras adaptativas do dispositivo, $IAR^r_{kr,tk}$, de DAC, convertendo-o em um novo conjunto de regras em $IAR^r_{kr,tk+1}$. As regras em $IAR^r_{kr,tk}$ (r=1, .., m) são da forma iar^r = (iba, ba^r, ci^r, s^r, cj^r, z^r, aa^r, iaa), significando que em resposta a algum estimulo $s^r \epsilon S^r$ ela executa inicialmente o subgrupo de mensagens anterior iba, em seguida a regra adaptativa ar^r = (ba^r,ci^r, s^r, cj^r, z^r, aa^r), e por fim o subgrupo de mensagens posterior iaa. A regra adaptativa ar^r é executada da mesma forma definida em (NETO, 2001).

3.4 Tipos de comunicação – Camada CC

A comunicação entre dois dispositivos deve ser composta, portanto, de mensagens em formatos pré-definidos, segundo o padrão estabelecido no protocolo de comunicação.

Podemos distinguir três tipos de comunicações na camada CC. O primeiro é a comunicação entre dois dispositivos do DAC, utilizando o mecanismo MGC como intermediário. Na figura 3.2 temos a representação do envio desse tipo de mensagem, “Msg(1,2)”, do dispositivo 1 para o dispositivo 2, passando pelo mecanismo MGC.

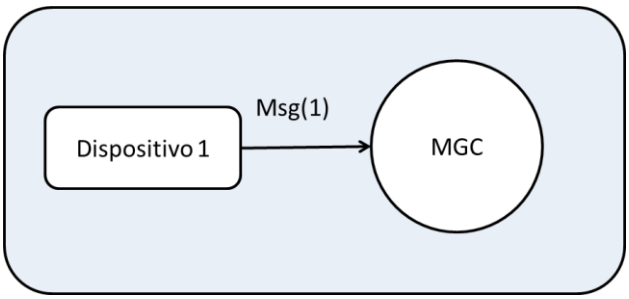
Figura 3.2- Exemplo de comunicação entre dois dispositivos distintos



Fonte: Autor

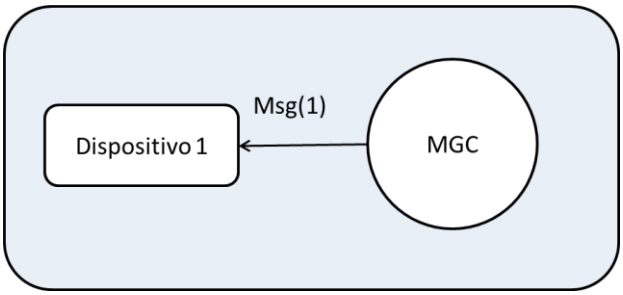
O segundo tipo é o envio de mensagem de um dispositivo do DAC para o mecanismo MGC, conforme ilustrado na figura 3.3. O terceiro tipo é o envio de mensagem do mecanismo MGC a um dispositivo do DAC, conforme ilustrado na figura 3.4.

Figura 3.3- Exemplo de comunicação entre um dispositivo e o MGC.



Fonte: Autor

Figura 3.4- Exemplo de comunicação entre MGC e um dispositivo.



Fonte: Autor

3.5 Mensagens padrão do protocolo de comunicação

A seguir, são apresentadas as descrições das mensagens do protocolo de comunicação, que podem ser classificadas em quatro categorias. A primeira, denominada inicial, é composta de funções que são executadas no inicio da cooperação dos dispositivos adaptativos, preparando o dispositivo para interagir com os demais e com o dispositivo de controle MGC. A segunda, denominada categoria de informação, engloba mensagens que fornecem as informações da configuração do dispositivo. A terceira, denominada alteração de configuração, se referem às mensagens com característica de alterar a configuração do dispositivo, possibilitando a inclusão e exclusão de regras. A quarta, denominada semáforo, tem a finalidade de gerenciar a comunicação, evitando mais de uma comunicação simultaneamente. A quinta, denominada de finalização, é responsável em limpar todas as associações efetuadas pela primeira categoria.

Uma mensagem pode ter uma das seguintes composições:

- Mensagem (emissor, destinatário, [lista de parâmetros]).
  Exemplo apresentado na figura 3.2.

- Mensagem (emissor, [lista de parâmetros]).
  Exemplo apresentado na figura 3.3.

- Mensagem (destinatário, [lista de parâmetros]).
  Exemplo apresentado na figura 3.4.

Quando existem parâmetros na mensagem, eles sempre são do tipo numérico (número natural).

A categoria inicial tem como finalidade efetuar as associações das propriedades do dispositivo, necessárias para utilização nas demais mensagens. Algumas mensagens são equivalentes aos procedimentos em uma linguagem de programação de alto nível. Outras mensagens são equivalentes às funções de uma linguagem de programação, retornando um valor inteiro.

Nesta seção, apresentam-se as mensagens padrões do protocolo, sem apresentar os detalhes, como os parâmetros e valores de retorno. No apêndice A há o detalhamento dessas informações.

3.5.1 Categoria Inicial

Essa categoria é composta de mensagens que preparam os dispositivos para inicio da simulação.

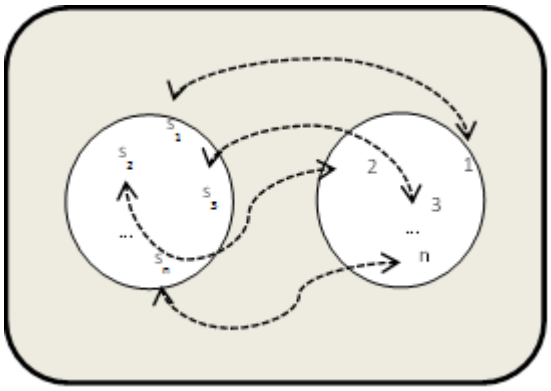
Inicial-Atribui-Identificador-Dispositivo

Atribui a um dispositivo do DAC, emissor da mensagem, um número de identificação único.

Inicial-Associação-Estímulos-Entrada-Dispositivo

Efetua uma associação dos seus elementos s_v^r do conjunto S^r ($s_v^r \in S^r$, $v \geq 0$), através de uma relação biunívoca, a um elemento de um conjunto de números naturais. Dessa forma, cada elemento do conjunto de eventos s_v^r pode ser referenciado através do número natural ao qual está associado, como exemplo o número 1, 2, etc., conforme ilustrado na figura 3.5.

Figura 3.5- Exemplo de associação dos eventos a um conjunto de números



Fonte: Autor

Inicial-Associação-Eventos-Saida-Dispositivo

Similar à mensagem anterior, associa cada elemento z_v^r do conjunto NA^r ($z_v^r \in NA^r$, v≥0), através de uma relação biunívoca, a um elemento de um conjunto de números naturais. Assim, cada elemento do conjunto de eventos de NA^r pode ser referenciado através do número natural ao qual está associado, como por exemplo 1, 3, etc.

Inicial-Associação-Configuração-Dispositivos

Analogamente, cada configuração $c^r_{kr,tk}$ ($c^r_{kr,tk} \in C^r_{kr,tk}$ kr≥ 0, tk≥0), do dispositivo emissor, é associada a um número natural através de uma relação biunívoca.

Inicial-Associação-Regras-Dispositivo

Cada regra $iar^r \in IAR^r_{kr,tk}$ ($r > 0$, $tk \geq 0$,$kr \geq 0$) é associada a um número natural através de uma relação biunívoca.

Inicial-Associação-Ação adaptativa anterior-Dispositivo

Cada ação adaptativa anterior do dispositivo é associada a um número natural através de uma relação biunívoca.

Inicial-Associação-Ação adaptativa posterior-Dispositivo

Cada ação adaptativa posterior do dispositivo é associada a um número natural através de uma relação biunívoca.

Inicial-Associação-Função Comunicação

Cada função de comunicação dos dispositivos é associada a um número natural através de uma relação biunívoca.

3.5.2 Categoria de informação

Essa categoria é composta de mensagens que fornecem informações sobre os dispositivos.

Informa-Situação

Essa mensagem retorna a configuração do dispositivo destinatário, ligado ou desligado (ON ou OFF, respectivamente).

Identificação-Regra

Retorna o número natural associado à regra, do dispositivo destinatário, que satisfaz as condições informadas pelos parâmetros.

Identificação-Estímulo-Atual

Retorna o número natural associado ao estímulo de entrada, do dispositivo destinatário.

Identificação-Configuração-Atual

Essa mensagem permite que o dispositivo, emissor da mensagem, solicite ao dispositivo destinatário o número associado à sua configuração atual.

3.5.3 Categoria de alteração de configuração

Configura-Situação

Essa função atribui ao dispositivo a sua situação de ligado ou desligado conforme o parâmetro informado, respectivamente ON ou OFF.

Novo-Estado

Essa mensagem permite que o dispositivo, emissor da mensagem, solicite ao dispositivo receptor que selecione uma nova configuração $c^{r}_{kr,tk}$ de $C^{r}_{kr,tk}$,

que ainda não esteja sendo utilizada em nenhuma regra. É similar à função geradora gn das funções adaptativas. O retorno da mensagem é o número natural associado à respectiva configuração.

Gera-Estímulo

O dispositivo emissor solicita ao dispositivo receptor simular o estímulo de entrada, do seu conjunto S^r de estímulo, associado ao número indicado pelo parâmetro.

Criação-Regra

Essa mensagem, cria uma regra no dispositivo destinatário da mensagem com os respectivos parâmetros informados. Ao se criar a regra, o dispositivo associa-a a um número natural ainda não utilizado em nenhuma regra.

Exclusão-Regra

Essa mensagem exclui a regra, do dispositivo destinatário, cujo número associado seja igual ao parâmetro informado.

3.5.4 Categoria de semáforo

Essa categoria é composta de mensagens utilizadas para sincronização e controle, permitindo apenas uma comunicação simultaneamente.

Inicio-comunicação-entre-dispositivos

Mensagem para iniciar a comunicação entre dois dispositivos. O mecanismo MGC gerencia essas chamadas para garantir que não ocorram duas ou mais comunicações simultaneamente.

Fim-comunicação-entre-dispositivos

Mensagem para finalizar a comunicação entre dois dispositivos.

3.5.5 Categoria Finalização

Essa categoria contém a mensagem responsável em finalizar o processo de interação entre os dispositivos. A camada MGC envia mensagem a todos os dispositivos, tornando-os inativos. Todas as informações relativas às configurações dos dispositivos são gravadas e o MGC torna-se inativo.

Término

Finaliza o processo de comunicação entre os dispositivos e a camada MGC, eliminando as informações carregadas pelas mensagens da categoria inicial.

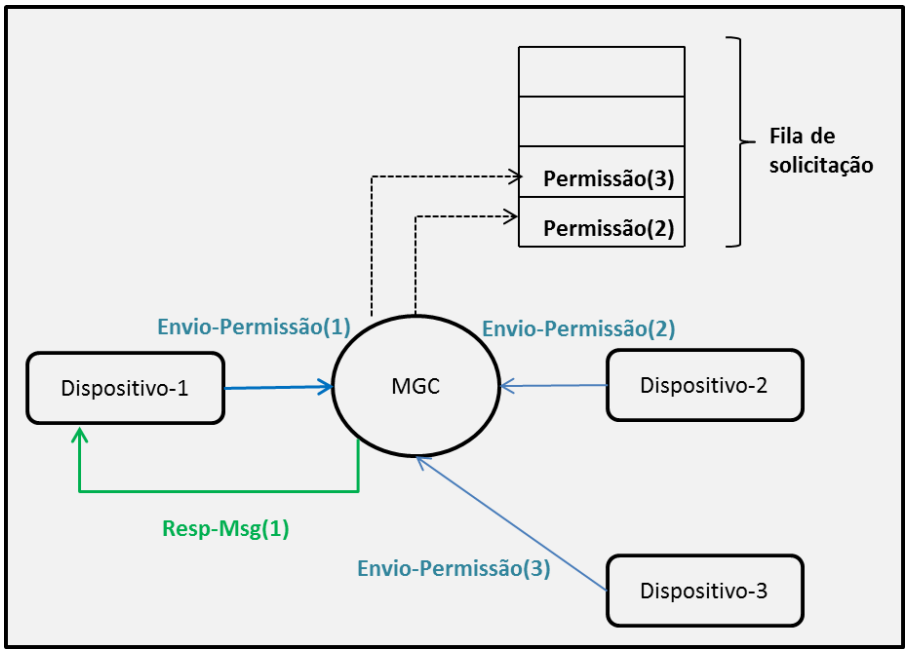
3.6 Mecanismo de Gerenciamento e Controle (MGC)

O mecanismo de Gerenciamento e Controle das interações entre os dispositivos é responsável por não permitir mais de uma comunicação simultânea entre dois dispositivos quaisquer.

Os pedidos de permissão de comunicação da camada CC são controlados através de uma fila de solicitação do tipo FIFO (“First In- First Out”).

A figura 3.6 ilustra o controle e gerenciamento de várias solicitações para execução de comunicação entre dispositivos. No exemplo ilustrado na figura temos a representação de 3 solicitações de comunicação. Entretanto como o mecanismo MGC permite apenas uma execução por vez, somente a solicitação do dispositivo 1 é executada, enquanto as demais aguardam em uma fila de espera.

Figura 3.6- Exemplo de gerenciamento de 3 solicitações de mensagens.



Fonte: Autor

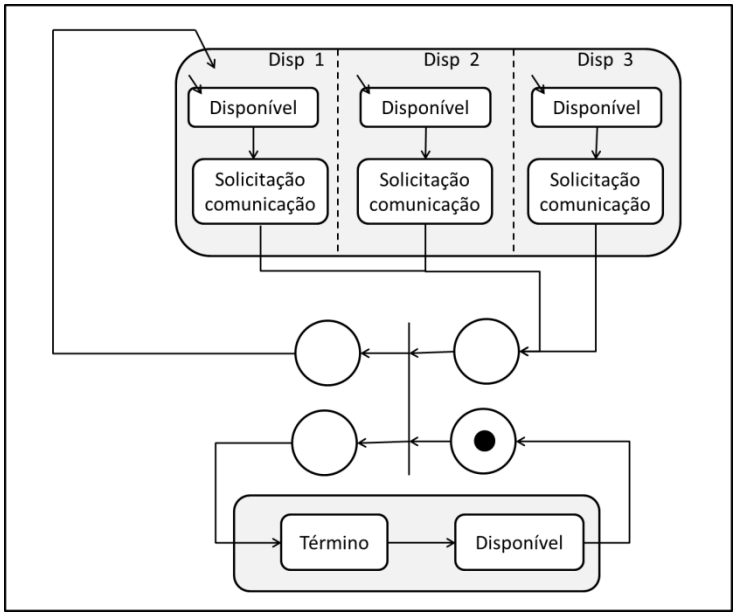
O mecanismo MGC pode ser representado por vários formalismos, se houver a necessidade de sua representação na modelagem do problema ao qual foi aplicado o DAC, como a Máquina de estados finitos, Máquina de Moore, Redes de Petri, Statecharts Adaptativos, etc. Entretanto, se as questões relacionadas ao mecanismo MGC não forem importantes na modelagem, não há necessidade de sua visualização, fazendo parte apenas da arquitetura de eventual implementação sistêmica do modelo. Dependendo da quantidade de dispositivos envolvidos no DAC, a utilização de alguns formalismos pode não ter uma representação adequada, devido às limitações do formalismo em particular utilizado.

Entretanto, apenas para ilustrar uma das possibilidades, a seção seguinte apresenta a utilização dos Statecharts adaptativos sincronizados (SANTOS, 1997) para modelar o mecanismo MGC.

3.7 Statecharts adaptativos sincronizados como Mecanismo de Gerenciamento e Controle (MGC)

Na utilização dos Statecharts adaptativos sincronizados, representamos em um *statechart* todos os dispositivos do DAC, um em cada estado ortogonal do estado do primeiro nível. No exemplo ilustrado na figura 3.7, temos três dispositivos representados no primeiro nível do Statecharts. O segundo *statechart* representa o recurso de comunicação.

Figura 3.7- Statechart sincronizado como MGC

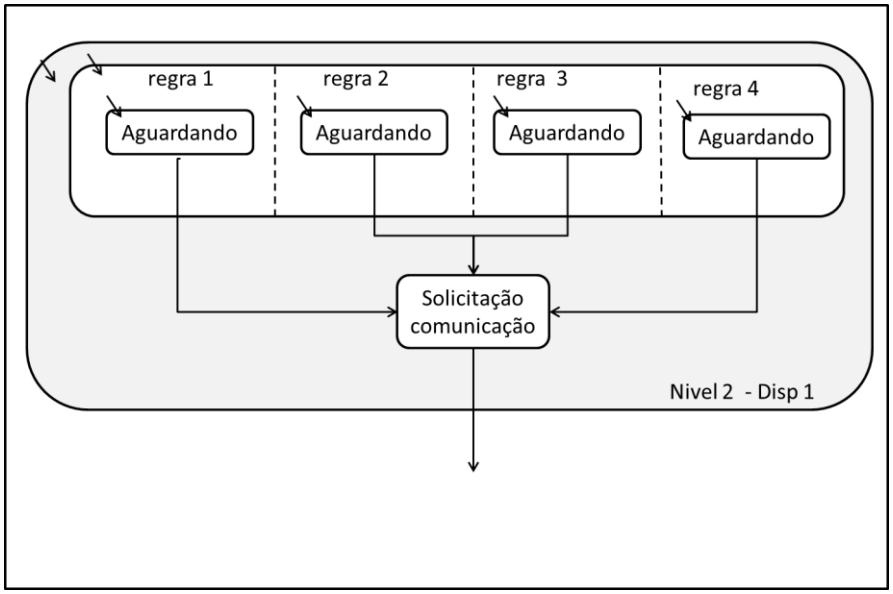


Fonte: Autor

A Rede de Petri sincroniza o pedido de comunicação com a disponibilidade do recurso, garantindo que nenhuma outra solicitação será executada enquanto a comunicação corrente não for finalizada

Cada estado, denominado “disponível”, representa, no segundo nível do Statecharts todas as possibilidades de comunicação, na qual o respectivo dispositivo é o emissor, presentes nas regras do DAC. A figura 3.8 ilustra um exemplo onde um dispositivo contém quatro regras onde há comunicação entre dispositivos.

Figura 3.8- Dispositivo contendo 4 regras com comunicação



Fonte: Autor

3.8 Exemplo 1 – DAC com Autômato Finito e Tabela de Decisão

Para ilustrar o funcionamento dos Dispositivos Adaptativos Cooperantes (DAC), apresentamos nesta seção um exemplo composto de dois dispositivos: uma tabela de decisão e um autômato finito. Neste exemplo, o autômato finito (AF) possui regras que promovem a troca de mensagens de comunicação com a tabela de decisão (DT), cuja interpretação resulta em alteração do seu conjunto de regras.

O autômato finito, do DAC, tem a capacidade de reconhecer cadeias compostas da concatenação de duas subcadeias, a primeira contendo caracteres alfabéticos (representado por γ) e a segunda cadeia formada por dígitos (representado por d). As duas subcadeias podem ser vazias. O símbolo que marca o final de cadeia de caractere é representado por "├". Simbolicamente podemos dizer que a linguagem reconhecida pelo autômato é: $L(FS) = \{ \gamma^m d^n \vdash \mid (m \geq 0, n \geq 0)\}$.

À medida que o autômato reconhece um símbolo válido, ele envia mensagens à tabela de decisão, a qual, ao executar as instruções, adiciona a regra de

reconhecimento do respectivo símbolo equivalente. A tabela de decisão é construída de tal maneira que ela representa o reconhecimento das cadeias já tratadas no autômato finito.

No DAC descrito, temos um dispositivo com a capacidade de reconhecimento de alguma informação (padrão), enquanto que o outro dispositivo representa o conhecimento adquirido pelo primeiro. Neste exemplo, a capacidade de modificação das regras do segundo dispositivo está implícita nas funções de comunicação do primeiro dispositivo.

A seguir apresentamos as definições e um exemplo ilustrativo de interação entre os dispositivos. Lembrando a definição apresentada, temos que cada elemento do conjunto de dispositivos é identificado com um número. Para facilitar a compreensão, vamos utilizar no exemplo, mnemônicos para representar os números de identificação dos dispositivos: TABELA (dispositivo 1), AUTOMATO (dispositivo 2) e MGC (Gerenciador de mensagens).

Dispositivo 1 - tabela de decisão

A seguir são apresentamos as informações relativas à configuração inicial da tabela de decisão.

C^1 = { CA, CB, CC, CD}

Associação padrão:

- Inicial-Associação-Configuração-Dispositivos (TABELA, CA) : 1
- Inicial-Associação-Configuração-Dispositivos (TABELA, CB) : 2
- Inicial-Associação-Configuração-Dispositivos (TABELA, CC) : 3
- Inicial-Associação-Configuração-Dispositivos (TABELA, CD) : 4

c^1_0 = { CA }

NA^1 = { OK, NOT_OK}

Associação padrão:

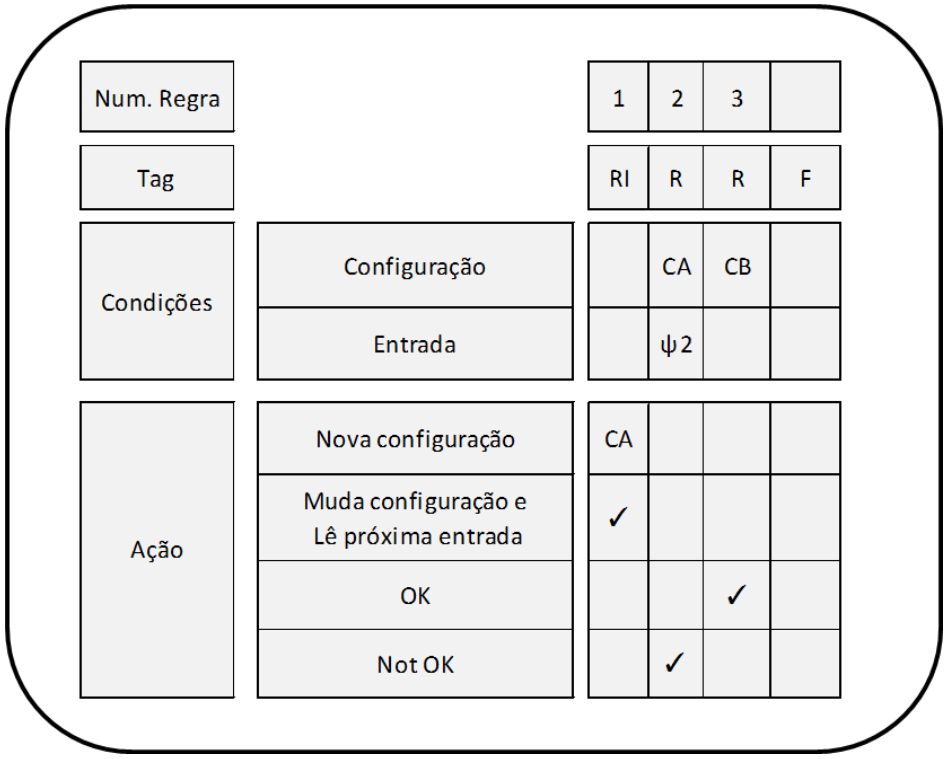
- Inicial-Associação-Eventos-Saida-Dispositivo (TABELA, OK) : 5
- Inicial-Associação-Eventos-Saida-Dispositivo (TABELA, NOT_OK) : 10

S^1 = { carac_alfa, carac_digito,símbolo_fim_cadeia }

- Inicial-Associação-Eventos-Entrada-Dispositivo (TABELA, carac_digito) : 1
- Inicial-Associação-Eventos-Entrada-Dispositivo (TABELA, carac_alfa) : 2
- Inicial-Associação-Eventos-Entrada-Dispositivo (TABELA, fim_cadeia) : 3

Graficamente, a tabela de decisão pode ser representada como a figura 3.9.

Figura 3.9- Configuração inicial da tabela de decisão.



Fonte: Autor

Regras:

AR^1 = { $regra^1[1]$, $regra^1[2]$, $regra^1[3]$, $regra^1[4]$, $regra^1[5]$, $regra^1[6]$, $regra^1[7]$, $regra^1[8]$, $regra^1[9]$}

- regra^1[1]: (∅, ∅, ∅, ∅, CA, ∅, ∅, ∅, ∅, ∅)
- regra^1[2]: (∅, ∅, ∅, CA, ψ2, ∅, reject, ∅, ∅, ∅)
- regra^1[3]: (∅, ∅, ∅, CB, ∅, ∅, accept, ∅, ∅, ∅)

onde ψ2 é uma entrada não válida, ou seja, ψ2 ∉ S^1.

Dispositivo 2 - Autômato

A seguir são apresentadas as informações relativas à configuração inicial do autômato, dispositivo 2. A configuração inicial do autômato está representada na figura 3.10.

$AD_k = (C^2_k, IAR^2_k, S^2_k, \mathbf{c^2_k}, A_k^2 , NA^2, BA^2, AA^2, IBA, IAA)$

Suas configurações iniciais (tk=0, k1=0)

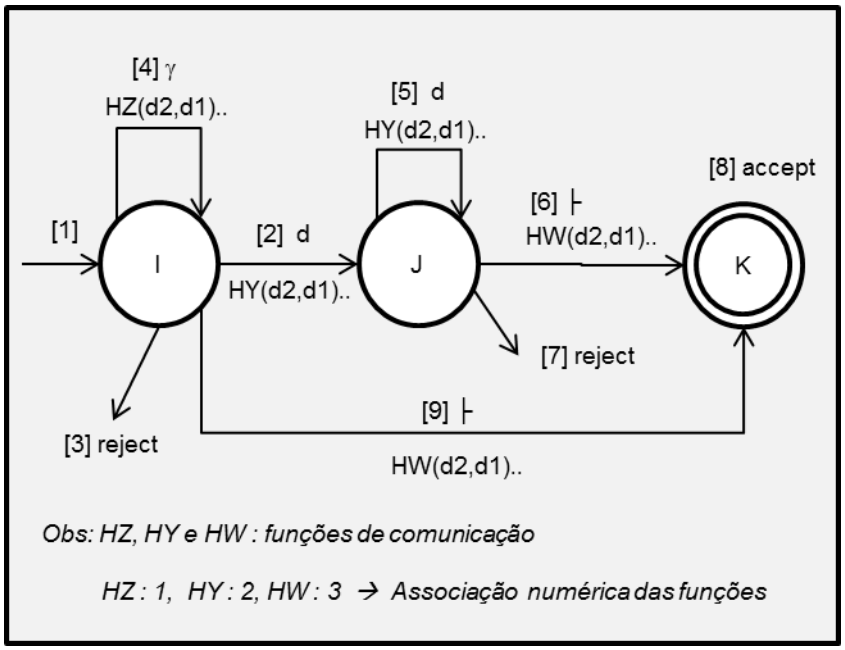
$AD^2_{0,0} = (C^2_0, IAR^2_0 , S^2_0 , \mathbf{c^2_0}, A^2_0 , NA^2, BA^2, AA^2, IBA, IAA)$

$C^2_0 = \{ I, J, K \}$

- Inicial-Associação-Configurações-Dispositivos (AUTOMATO, I): 1
- Inicial-Associação-Configurações-Dispositivos (AUTOMATO, J): 2
- Inicial-Associação-Configurações-Dispositivos (AUTOMATO, K): 3

$C^2_0 = \{ I \}$

Figura 3.10- Configuração inicial do autômato.



Fonte: Autor

A^2 = { K }

NA^2 = { reject, accept}

- Inicial-Associação-Eventos-Saida-Dispositivo (AUTOMATO, reject) : 1
- Inicial-Associação-Eventos-Saida-Dispositivo (AUTOMATO, accept) : 2

S^2 = { γ, *d*, ├ }

- Inicial-Associação-Eventos-Entrada-Dispositivo (AUTOMATO, γ) : 1
- Inicial-Associação-Eventos-Entrada-Dispositivo (AUTOMATO, *d*) : 2
- Inicial-Associação-Eventos-Entrada-Dispositivo(AUTOMATO, ├) : 3

IBA = { HZ, HY, HW }

- Inicial-Associação-Função Comunicação (AUTOMATO, HZ) : 1
- Inicial-Associação- Função Comunicação (AUTOMATO, HY) : 2
- Inicial-Associação- Função Comunicação (AUTOMATO, HW) : 3

AR^2 = { $regra^2[1]$, $regra^2[2]$, $regra^2[3]$, $regra^2[4]$, $regra^2[5]$, $regra^2[6]$, $regra^2[7]$, $regra^2[8]$, $regra^2[9]$}

Onde:

$regra^2$ [1]: (∅, ∅, ∅, 1, ∅, ∅, ∅, ∅, ∅, ∅)

$regra^2$ [2]: (2(AUTOMATO, TABELA, MGC) , ∅, ∅, 1, 2, 2, ∅, ∅, ∅, ∅)

$regra^2$ [3]: (∅, ∅, ∅, 1, ψ1, ∅, 1, ∅, ∅, ∅)

$regra^2$ [4]: (1(AUTOMATO, TABELA, MGC), ∅, ∅, 1, 1, 1, ∅, ∅, ∅, ∅)

$regra^2$ [5]: (2(AUTOMATO, TABELA, MGC), ∅, ∅, 2, 2, 2, ∅, ∅, ∅, ∅)

$regra^2$ [6]: (3(AUTOMATO, TABELA, MGC) , ∅, ∅ ,2, 3, 3, ∅, ∅, ∅, ∅)

$regra^2$ [7]: (∅, ∅, ∅ ,2, ψ1, ∅, 1, ∅, ∅, ∅)

$regra^2$ [8]: (∅, ∅, ∅, 3, ∅, ∅, 2, ∅, ∅, ∅)

$regra^2$ [9]: (3(AUTOMATO, TABELA, MGC), ∅, ∅, 1, 3, 3, ∅, ∅, ∅, ∅)

onde ψ1 é uma entrada não válida, ou seja, ψ1 ∉ S

As funções de comunicação identificadas por 1(AUTOMATO, TABELA, MGC), 2(AUTOMATO, TABELA, MGC) e 3(AUTOMATO, TABELA, MGC) representam as funções HZ, HY e HW contidas nas regras [2], [4], [5], [6] e [9]. Os símbolos, mnemônicos, dentro dos parênteses representam respectivamente a identificação do, emissor, destinatário e gerenciador da camada de comunicação.

Descrição da função de comunicação – HZ

Função de comunicação -HZ
***Parâmetros*:** Disp-Orig , Disp-dest, Disp-Controle
Variáveis v1, v2, v3, v4, v5, v6 : tipo inteiro

Inicio-comunicação-entre-dispositivos(Disp-Orig, Disp-dest , Disp-Controle*)*

*v2 = Novo_Estado(*Disp-Orig, Disp-dest*)*
*v1 = Estado_Atual(*Disp-Orig, Disp-dest*)*
*v5 = Identificação-regra(*Disp-Orig, Disp-dest*, Ø, Ø, v1, 2, v2, Ø, Ø, Ø)*
*v6 = Exclusão-regra(*Disp-Orig*,1,* Disp-dest*, v6)*
*v3 = Criacao_regra(*Disp-Orig, Disp-dest*, Ø, Ø, v1, 2, v2, Ø, Ø, Ø)*
*v4 = Criacao_regra(*Disp-Orig, Disp-dest*, Ø, Ø, v2, ψ2, Ø, 10, Ø, Ø)*

Fim-comunicação-entre-dispositivos(Disp-Orig, Disp-dest , Disp-Controle*)*

As duas primeiras mensagens são necessárias para garantir o controle e execução de apenas uma comunicação, simultaneamente. Como a regra que será criada já pode existir devido a outro reconhecimento, efetuamos a verificação e exclusão por meio da quinta e sexta mensagens. Verifica-se através da variável v5 a identificação da regra, se existir, e efetuamos a exclusão através da mensagem associada à variável v6. Se a regra não existir, v5 não tem nenhum valor associado, a mensagem de exclusão não tem efeito nenhum. A variável v3 está relacionada à nova regra criada, que associa o evento 2 (“carac_alfa”) da tabela de decisão a uma transição do estado atual para o novo estado, representado por v2, simulando o reconhecimento do “γ” pelo autômato. As duas últimas mensagens indicam à camada MGC que a comunicação foi finalizada, podendo-se iniciar outra comunicação, se houver.

Descrição da função de comunicação – HY

Função de comunicação - HY
Parâmetros: *Disp-Orig , Disp-dest, Disp-Controle*
Variáveis *v1, v2, v3, v4, v5, v6 : tipo inteiro*

Inicio-comunicação-entre-dispositivos (Disp-Orig, Disp-dest , Disp-Controle)

V2 = Novo_Estado(Disp-Orig, Disp-dest)
V1 = Estado_Atual(Disp-Orig, Disp-dest)
V5 = Identificação-regra(Disp-Orig, Disp-dest, Ø, Ø , v1, 1, v2, Ø, Ø, Ø)
V6 = Exclusão-regra(Disp-Orig, Disp-dest ,v6)
v3 = Criacao_regra(Disp-Orig, Disp-dest, Ø, Ø , v1, 1, v2, Ø, Ø, Ø)
v4 = Criacao_regra(Disp-Orig, Disp-dest, Ø, Ø , v2, ψ2, Ø, 10, Ø, Ø)

Fim-comunicação-entre-dispositivos (Disp-Orig, Disp-dest , Disp-Controle

A função HY é análoga à HX, solicitando a criação de uma regra associada ao evento 1, que representa um caractere digito.

Descrição da função de comunicação - HW

Função de comunicação - HW
Parâmetros: *Disp-Orig , Disp-Dest, Disp-Controle*
Variáveis *v1, v2 : tipo inteiro*

Inicio-comunicação-entre-dispositivos (Disp-Orig, Disp-dest , Disp-Controle)

v1 = Estado_Atual(Disp-Orig, Disp-Dest)
v2 = Criacao_regra(Disp-Orig, Disp-dest, Ø, Ø , v1, 3, 4, 5, Ø, Ø)

Fim-comunicação-entre-dispositivos (Disp-Orig, Disp-dest , Disp-Controle

A função de comunicação HW solicita a criação de uma nova regra no dispositivo destino, associado ao consumo do símbolo de final de cadeia (“fim_cadeia” associado ao número 3), no estado atual, passando para o estado 4 (número associado ao estado final “CD” da tabela de decisão). Como a configuração “CD” é de aceitação, o oitavo parâmetro é o “5”, indicador de aceitação.

3.8.1 Simulação da comunicação entre os dispositivos

A seguir estudaremos o comportamento da comunicação entre os dois dispositivos, na simulação do tratamento da entrada “γγ├” no dispositivo 2, o autômato finito (AF).

Figura 3.11- Configuração da Tabela após a primeira alteração nas regras.

Número da regra		1	2	3	4	5	6
Tag		IR	R	R	R	R	F
Condição	Configuração		CA		CA	CA	
	Entrada		ψ2		carac_alfa	ψ2	
Ação	Nova configuração	CA		CB	CC		
	Muda configuração e Lê próxima entrada	✓			✓		
	OK			✓			
	Not OK		✓			✓	

Fonte: Autor

Após o tratamento do primeiro elemento da cadeia de entrada, um “γ”, o autômato executa a regra [2], que envia mensagens, que quando interpretadas pela tabela de decisão resultam em inclusão de novas regras.

A função de comunicação HZ, executada anteriormente à regra [2] do autômato (AF), impacta na adição das regras [4] e [5] na tabela de decisão. A figura 3.11 ilustra a configuração da tabela de decisão, após a primeira comunicação do autômato.

Na tabela de decisão, a regra [4] representa a aceitação do “carac_alfa” que representa um caractere alfabético, da mesma forma que é representado no autômato finito por “γ”. Os conjuntos S^1 e S^2 são diferentes, mas há uma relação implícita entre eles.

Após o tratamento do segundo elemento da cadeia de entrada, “γ”, o autômato executa a regra [4], que possui a função de comunicação anterior HZ. Analogamente às alterações efetuadas na primeira execução, são adicionadas as regras [6] e [7] na tabela de decisão (TD). A figura 3.12 apresenta a configuração da tabela de decisão após o tratamento do segundo “γ” pelo autômato.

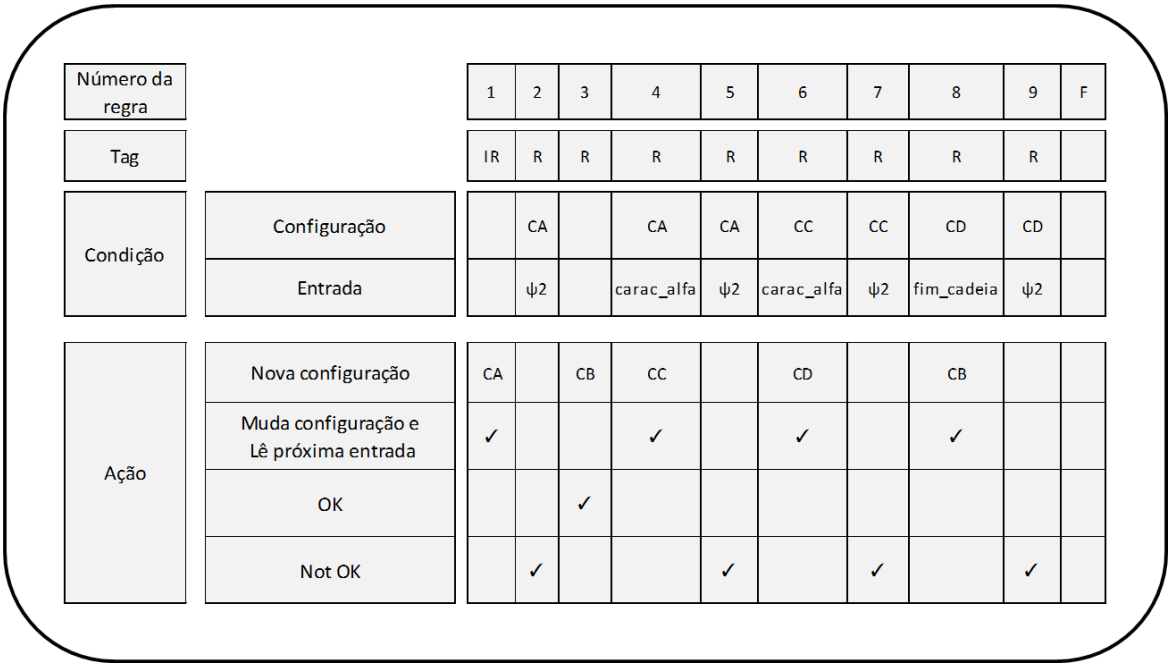
Figura 3.12- Configuração da Tabela após a segunda alteração nas regras.

Número da regra		1	2	3	4	5	6	7	F
Tag		IR	R	R	R	R	R	R	
Condição	Configuração		CA		CA	CA	CC	CC	
	Entrada		ψ2		carac_alfa	ψ2	carac_alfa	ψ2	
Ação	Nova configuração	CA		CB	CC		CD		
	Muda configuração e Lê próxima entrada	✓			✓		✓		
	OK			✓					
	Not OK		✓			✓		✓	

Fonte: Autor

Após o tratamento do último elemento da cadeia, identificador de final da cadeia, o autômato altera o comportamento da tabela, resultando na adição das regras [8] e [9], conforme ilustrado na figura 3.13

Figura 3.13- Configuração final da Tabela.



Número da regra		1	2	3	4	5	6	7	8	9	F
Tag		IR	R	R	R	R	R	R	R	R	
Condição	Configuração		CA		CA	CA	CC	CC	CD	CD	
	Entrada		ψ2		carac_alfa	ψ2	carac_alfa	ψ2	fim_cadeia	ψ2	
Ação	Nova configuração	CA		CB	CC		CD		CB		
	Muda configuração e Lê próxima entrada	✓			✓		✓		✓		
	OK			✓							
	Not OK		✓			✓		✓		✓	

Fonte: Autor

Após o término do tratamento da cadeia “γγ⊢” pelo autômato, a cadeia “carac_alfa-carac_alfa-fim_cadeia” é aceita na tabela de decisão com a execução das regras [1], [4], [6], [8] e [3].

Resumindo, a tabela de decisão possui a capacidade de reconhecer como entradas válidas apenas as sequências já reconhecidas pelo autômato, enquanto o autômato possui um conjunto de regras com a capacidade de aprender o reconhecimento de qualquer cadeia formada por caracteres alfabéticos seguida de uma cadeia formada por dígitos, ambas podendo ser a cadeia vazia.

Esse exemplo apresentado ilustra a capacidade dos dispositivos adaptativos cooperantes (DAC), onde um dispositivo possui a capacidade de aprendizado e o outro explicita o conhecimento já adquirido pelo primeiro dispositivo.

3.9 Exemplo 2 – DAC com Parser e Ontologia

Nesta seção é apresentado um exemplo de uma aplicação cuja modelagem pode ser composta de dois dispositivos, um “parser” e uma ontologia. Através de padrões sintático-semânticos o “parser” identifica possíveis relações semânticas entre dois conceitos, gerando uma comunicação com o outro dispositivo, a ontologia, resultando na adição de regras, representando a relação entre os conceitos.

Os padrões sintático-semânticos utilizados no dispositivo “parser” são baseados nos padrões apresentados nos trabalhos (HEARST, 1992) , tais como:

a) SN_0 tais como SN_1 {, SN_2, ...(e | ou) SN_i}

b) SN_0 {, SN,}* { , } e | ou SN_2

c) Tipos de SN: SN_1 {, SN_2, ,} (e | ou) SN_i

d) SN chamado de SN_2.

Trabalhos que apresentam a mesma técnica para construção de ontologia a partir de padrões sintáticos podem ser encontrados em (VELARDI, FARALLI e NAVIGLI, 2013) e (MORIN e CHRISTIAN, 2004).

Nas descrições apresentadas, “SN” são sintagmas nominais, “|” indica opções de escolha que podem ocorrer, “{}” indica uma lista de opções e “*” indica que a ocorrência é opcional.

O “parser” processa um “corpus”, extraindo o texto que seja formado por um dos padrões apresentados, identificando as relações semânticas. Quando as relações são identificadas, o “parser” envia uma mensagem à ontologia, que cria a respectiva regra para identificação da relação semântica.

Para ilustrar o exemplo, consideramos duas frases sobre copa do mundo.

Frase 1:

“Na copa do mundo, os visitantes viram vários jogadores de futebol tais como Messi, Iniesta e Cristiano Ronaldo”.

A frase 1 segue o padrão apresentado no item a, dessa forma o “parser” extrai as seguintes relações:

- hiperonímia(“Messi”,”jogador de futebol”),
- hiperonímia(“Iniesta”,”jogador de futebol”) e
- hiperonímia(“Cristiano Ronaldo”,”jogador de futebol”).

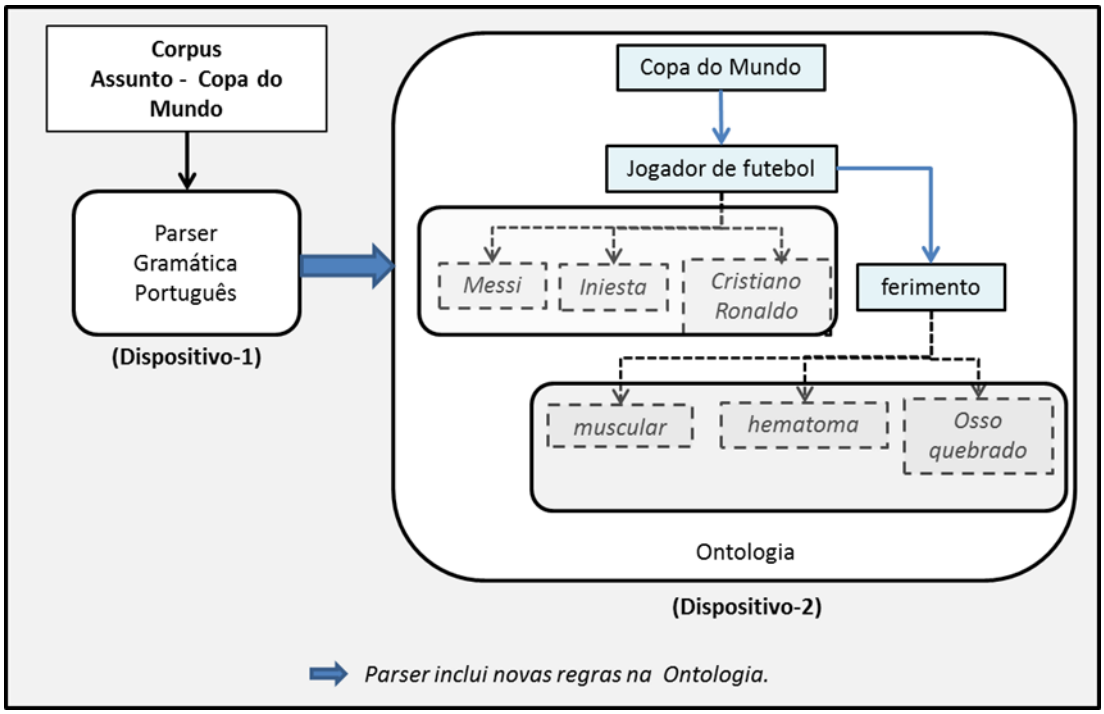
Frase 2:

“Os jogadores de futebol podem sofrer os tipos de ferimentos: muscular, hematoma e osso quebrado”.

A frase 2 segue o padrão apresentado no item c, de onde são extraídas as relações:

- hiperonímia(“muscular”,”ferimento”),
- hiperonímia(“hematoma”,” ferimento”) e
- hiperonímia(“osso quebrado”,” ferimento”).

Figura 3.14- Comunicação entre dois dispositivos



Fonte: Autor

A figura 3.14 ilustra o exemplo apresentado, indicando as novas relações que são incluídas na ontologia, após a identificação de um padrão por parte do "parser" e a sua respectiva comunicação de cooperação.

4 APLICAÇÃO – DISPOSITIVOS ADAPTATIVOS COOPERANTES

Com base nos dispositivos adaptativos cooperantes (DAC), desenvolveu-se uma aplicação composta de dois dispositivos.

O primeiro dispositivo é um autômato finito adaptativo e o segundo é uma tabela de decisão adaptativa que monitora o primeiro dispositivo, controlando seu funcionamento através de envio de mensagens, que alteram a configuração do autômato. A tabela de decisão efetua o seu monitoramento com base no histórico das configurações do Autômato Finito, ou seja, com base no histórico de seu comportamento do próprio autômato.

Em resumo, com base em um critério de decisão e o comportamento histórico do próprio autômato, a tabela de decisão modifica o comportamento do autômato.

Essa é uma característica que pode ser aplicada à várias classes de problemas. Uma classe, por exemplo, é a utilização de um dispositivo para executar um processo, enquanto o segundo tem a função de monitorar o ambiente, as variáveis que influenciam o sistema, e em função de alguma alteração dessas variáveis, efetuar uma modificação no comportamento do dispositivo executor, alterando a sua configuração.

Uma segunda classe é a utilização de um dispositivo para execução de uma atividade, enquanto o segundo dispositivo efetua inferência em informações históricas do comportamento do primeiro dispositivo, podendo modificar o comportamento do primeiro dispositivo através de alguns ajustes, representando um aprendizado com o histórico.

A aplicação é composta de dois dispositivos, o dispositivo executor (autômato) e o dispositivo monitor (tabela de decisão). Em cada ciclo de simulação o dispositivo monitor efetua o monitoramento do dispositivo executor, efetua-lhe uma modificação, alterando o seu conjunto de regras de forma indireta. O objetivo final dos dispositivos é atingir uma condição final do executor, que representa o objetivo a ser alcançado.

4.1 Autômato finito Adaptativo

O autômato finito adaptativo foi criado tomando como inspiração a formulação de jogos gerais, descritos no capítulo 2. Como em cada configuração temos uma quantidade grande de alternativas para novas configurações não é possível representar de forma explícita todas as regras e todas as combinações possíveis que um jogo pode ter, devido à explosão exponencial da combinação dos estados, um problema clássico dos Autômatos.

Assim, o Autômato tem a sua configuração alterada através de mensagens enviadas pela Tabela de Decisão, que a cada passo insere uma nova regra no Autômato.

A tabela de decisão analisa as possibilidades de alteração em função da configuração corrente do Autômato, do seu histórico e das regras do Autômato, descritas em linguagem GDL, que estão armazenadas em uma base de dados.

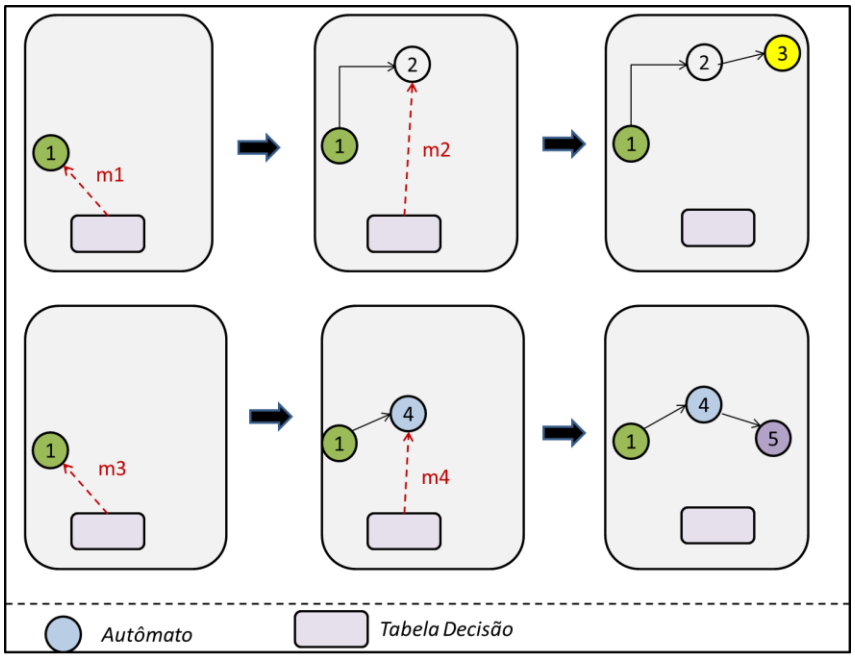
Para todas as possíveis regras que podem ser aplicadas em cada passo, a Tabela seleciona a melhor opção possível, segundo uma função de critério. Se no final houver mais de uma regra possível, seleciona-se apenas uma, de forma aleatória. Então a tabela cria uma regra de forma dinâmica, contendo funções de mensagens, as quais criarão de forma indireta uma regra adaptativa no autômato. Assim, ao escolher a única alternativa, a tabela de decisão comunica o Autômato, que em resposta cria uma regra, que é executada, modificando a sua configuração. Ao término da execução de cada regra no Autômato, ela se auto-exclui, iniciando-se novo ciclo na Tabela de Decisão.

A regra criada no Autômato a cada passo representa a alteração do Autômato para a configuração mais adequada, selecionada pela Tabela de Decisão, segundo a função de critério, para se atingir ou se aproximar de um estado final.

A figura 4.1 representa as configurações do Autômato em dois momentos diferentes, quando a tabela de decisão é submetida aos mesmos estímulos (E1 e E2), porém envia mensagens diferentes ao Autômato (m1, m2, m3 e m4), devido ao fato de os históricos serem diferentes. Na primeira representação o Autômato

ficou com as configurações 1, 2 e 3, enquanto na segunda representação ficou com as configurações 1, 4 e 5.

Figura 4.1- Comunicação dos dispositivos.



Fonte: Autor

Conforme apresentado no capitulo 2, existe uma linguagem chamada GDL, utilizada para definir uma classe de jogos.. O dispositivo Autômato do nosso modelo é representado na forma da linguagem GDL, cujas informações são armazenadas em uma base de dados. Para informações e detalhes relativos à linguagem GDL, recomenda-se o trabalho (GENESERETH; LOVE; PELL, 2005)

Dessa forma, conforme apresentado no capitulo 2, a base de dados com as informações na linguagem GDL da definição do autômato, possui as seguintes informações:

- Regras de Configuração
- Regras da entrada

- Informações iniciais
- Regras de mudança de configuração
- Definição de conceitos
- Regras de validação
- Regra de pontuação
- Condições de término

As configurações iniciais são utilizadas para configurar as condições iniciais do autômato. Cada estimulo de entrada é validado através das regras de entrada.Na sequencia é necessário validar a entrada sob as condições correntes, através das regras de validação.

Após todas validações, verifica-se a configuração atual é uma condição final, através das regras de condições de término. Quando a configuração atual for uma condição final, pode-se avaliar através das regras de pontuação se o objetivo foi ou não atingido.

Se a entrada foi validada e a configuração atual não é um estado final, verificam-se as regras que podem ser aplicadas, através das regras de configuração e regras de mudança de configuração. Se houver apenas uma regra permitida, ela é efetuada. No caso de haver mais de uma opção, a escolha do autômato sozinho é arbitrária.

4.2 Tabela de Decisão Adaptativa

O segundo dispositivo do nosso modelo é definido através de uma tabela de decisão adaptativa, que será identificada a partir de agora, até o final do capítulo como tabela T.

A Tabela Adaptativa, em cada ciclo de execução, verifica quais as possíveis regras que o autômato pode executar na configuração corrente, através das informações da base de dados (regras GDL). Para cada possível regra, ela associa um valor, maior que zero, aplicando uma função de critério. Esse valor está relacionado com a possibilidade de essa regra levar o Autômato para uma configuração final de aceitação, ou seja, quanto maior o valor, maiores são as

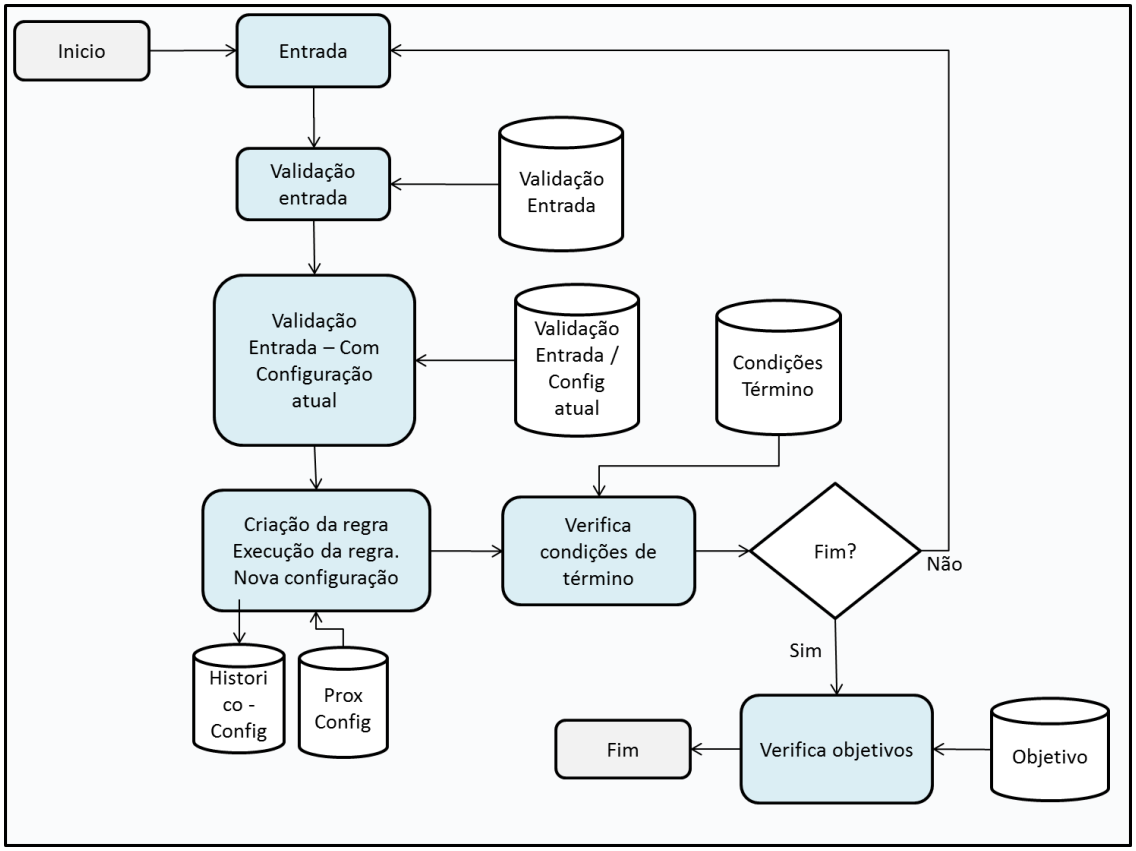
possibilidades de o autômato atingir um estado final, após um número finito de alterações. Esse valor é calculado através de uma função que analisa as regras que indicam as condições de término, ou seja, as condições que levam o autômato para uma configuração final de aceitação.

Uma configuração final pode representar sucesso ou não, dependendo de o sistema vencer ou não a partida, conforme seja indicado pela regra de pontuação.

4.3 Fluxograma

A figura 4.2 ilustra o fluxo da execução da aplicação, com o recurso da função de avaliação da tabela T desabilitado, ou seja, a execução arbitrária do autômato apenas.

Figura 4.2- Fluxograma- Sem utilização da função critério.



Fonte: Autor

O ciclo se inicia com uma entrada, que é analisada. O segundo passo é a validação da entrada, sob as condições correntes do dispositivo, porque algumas entradas são válidas sob algumas regras. Para ajudar a compreensão, quando aplicada à execução de um jogo essa análise caracteriza a validação de um lance, conforme a configuração do jogo. No jogo de damas, por exemplo, não se pode mover uma peça para uma posição do tabuleiro que já esteja ocupada por uma peça da mesma cor.

Após a validação da entrada, temos a criação da regra na Tabela, a sua execução e como consequência a criação da regra no autômato, passando-o para a nova configuração.

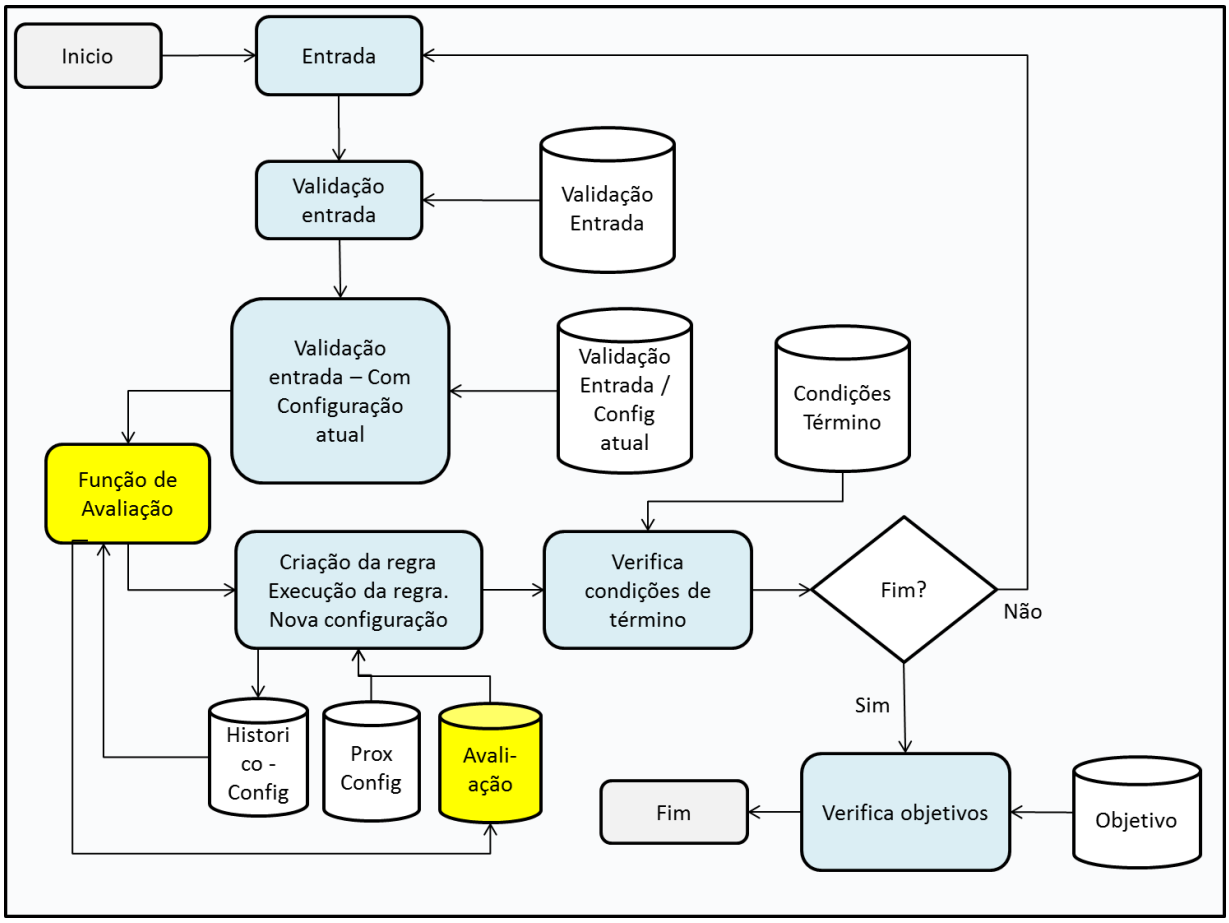
Em seguida, verifica-se se alguma condição de término foi atingida. Em caso negativo, inicia-se o ciclo. Em caso afirmativo, avaliam-se o objetivo para verificar se houve sucesso, atingindo uma configuração de êxito na busca do objetivo.

A figura 4.3 ilustra o fluxo contendo a função de avaliação. Nessa situação, a Tabela T seleciona a melhor opção para a próxima configuração, entre as alternativas possíveis, conforme as informações históricas e a função de avaliação.

4.4 Função de Avaliação

A função de avaliação pode ser considerada como uma função que avalia a configuração atual do dispositivo e as possíveis regras que podem ser executadas no momento, tentando escolher a melhor alternativa. A melhor alternativa é escolhida tendo como objetivo atingir um estado final com êxito.

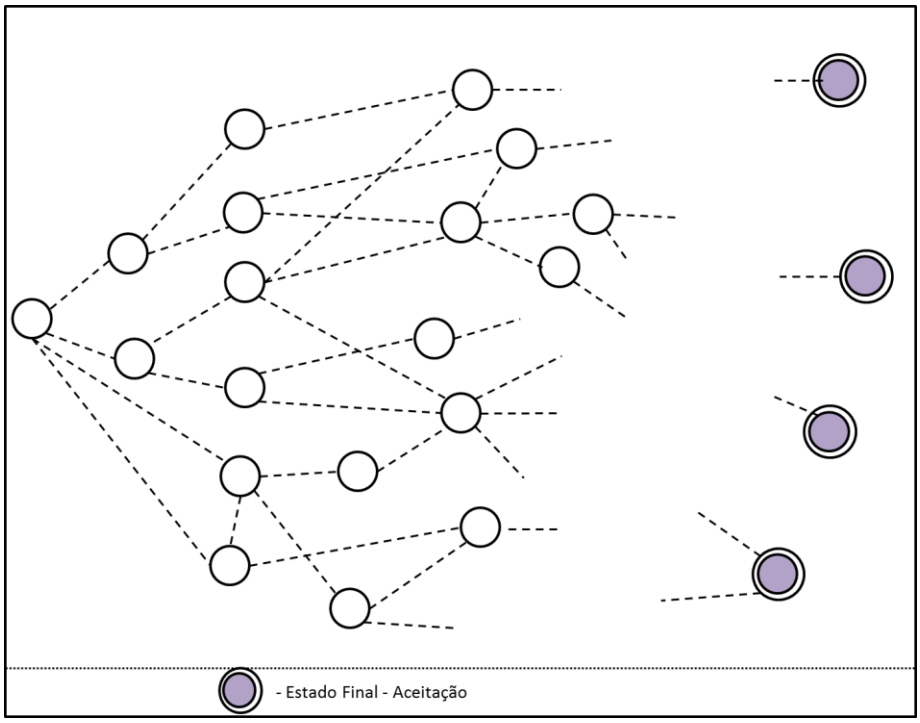
Figura 4.3- Fluxograma- Com utilização da função critério.



Fonte: Autor

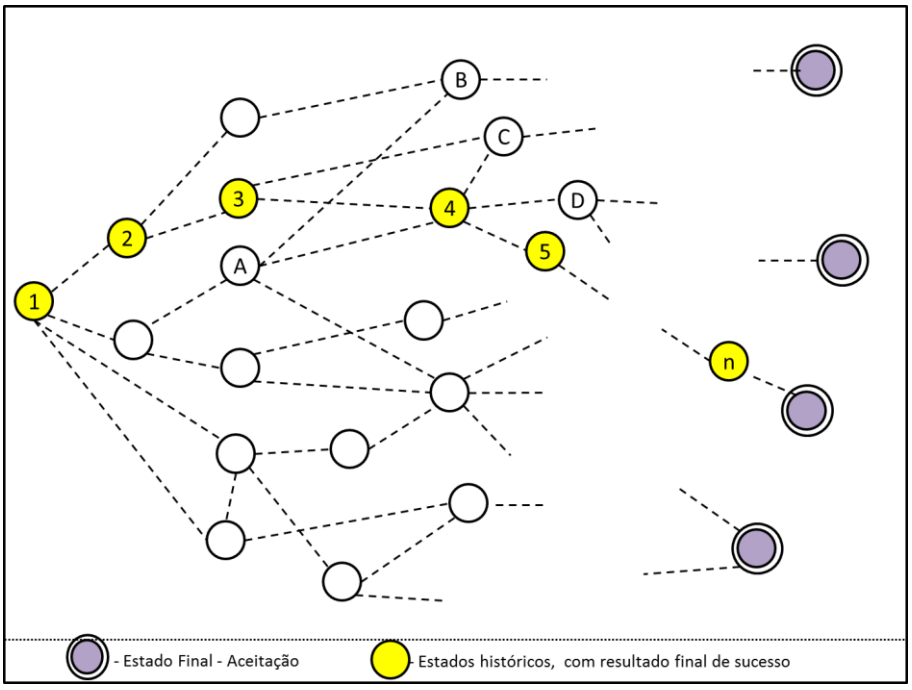
Graficamente ela pode ser representada nas figuras 4.4 e 4.5. A figura 4.4 apresenta a opção de um autômato sem a utilização da função de avaliação, ou seja, a escolha aleatória das possíveis regras. A figura 4.5 ilustra em amarelo as configurações que levaram a uma condição final (histórico). Essas informações auxiliam a função de avaliação a escolher as configurações que sejam amarelas ou que convirjam o máximo possível para uma configuração amarela. A figura **4.5** ilustra que, com a utilização da função de avaliação, se a configuração do autômato for igual à configuração “3”, a próxima configuração a ser escolhida será a “4”, e não a “C”.

Figura 4.4- Configurações possíveis.



Fonte: Autor

Figura 4.5- Configurações possíveis utilizando função de avaliação.

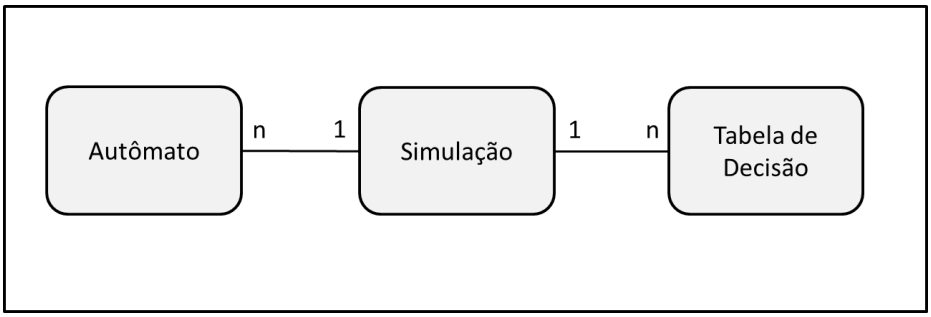


Fonte: Autor

4.5 Arquitetura

A aplicação foi desenvolvida utilizando a linguagem Java (“SE Development Kit – JDK 7”), com banco de dados *Access* 2010 da *Microsoft*, para armazenamento das informações, na plataforma *Windows 7*.

Figura 4.6- Principais classes da Aplicação.



Fonte: Autor

Foram construídas três classes. A primeira, denominada “Autômato”, utilizada para representar o funcionamento do autômato. A segunda, chamada de “Tabela de Decisão”, é a responsável pela seleção da melhor regra a ser executada em cada ciclo. Para viabilizar a execução do modelo, temos a classe “Simulação”, responsável em efetuar uma simulação do modelo composto de objetos das outras duas classes.

5 DISPOSITIVOS ADAPTATIVOS COOPERANTES – APLICAÇÃO EM JOGOS GGP

Os sistemas de jogos GGP (GGP em inglês, “General Game Playing”) tem como principal objetivo jogar jogos sem conhecimento prévio das suas regras. As regras desses jogos são descritas em uma linguagem GDL (em inglês, “General Definition Game”). O principal desafio desses sistemas é ter a capacidade de compreender as regras do jogo apenas em tempo de execução.

Muitos trabalhos foram desenvolvidos em Inteligência Artificial relacionados a jogar um jogo especifico, como xadrez, damas ou jogo de cartas. Nesses sistemas, o projetista do programa elabora uma estratégia específica para um tipo de jogo, sendo de difícil aproveitamento em outro jogo. As estratégias elaboradas são dependentes do jogo e do conhecimento do projetista sobre esse jogo em particular.

Entretanto, os sistemas de jogos GGP tem a finalidade de jogar uma variedade de jogos, mesmo um jogo que ainda nunca tenha sido jogado.

Os sistemas de jogos GGP têm como principal característica a capacidade de aprender estratégias de um jogo de forma dinâmica e autônoma.

Os sistemas GGP possuem duas principais características, a capacidade de se configurar de forma dinâmica e autônoma, conforme as regras do jogo, e a sua capacidade de aprender estratégia para o jogo, com base na experiência e resultados dos jogos realizados (histórico).

O sistema GGP desenvolvido, o qual será referenciado a partir desse ponto como Adap-GGP, utilizou a arquitetura apresentada no capitulo anterior. O dispositivo autômato representa uma aplicação GGP, enquanto a tabela de decisão é responsável pelo gerenciamento do autômato, avaliando a melhor estratégia, utilizando as regras do jogo e o histórico dos jogos já realizados.

Assim, conforme aumenta a quantidade de jogos realizados pela aplicação, aumentam correspondentemente as informações do histórico, que servem de

base para os cálculos das estratégias. Porém as estratégias são realizadas com base nas informações das regras dos jogos e não de forma estatística como grande parte das aplicações encontradas na literatura.

5.1 Estratégias

A solução utilizada na elaboração das estratégias tem como base as informações das regras e dos resultados históricos dos jogos, traduzidos pela aplicação em forma de conhecimento para tomada de decisão. A aplicação desenvolvida possui quatro possibilidades de estratégias, utilizadas na escolha dos lances em cada lance do jogo, baseadas no seu conhecimento histórico.

Para efetuar condições de comparação e análise da eficiência das estratégias, foram adicionadas mais duas estratégias. A primeira tem como objetivo ter uma estratégia sem a influência da Tabela de Decisão, tornando aleatória a execução do autômato. A segunda estratégia adicionada é uma estratégia estatística.

A estratégia de jogo é equivalente à função avaliação da arquitetura geral, apresentada no capítulo anterior.

As estratégias elaboradas com base no histórico são baseadas em duas ideias principais. A primeira consiste em efetuar os lances de forma que a configuração atual do jogo fique o mais próximo possível de uma configuração de sucesso, ou seja, uma configuração que o dispositivo teve em um jogo, cujo resultado final foi a vitória. O módulo de estratégias verifica se em algum jogo, entre os jogos com resultado positivo no final, houve uma configuração igual à configuração atual. Quando há essa coincidência, repete-se o lance que o jogo histórico efetuou, tentando dessa forma imitar alguma sequência de jogo já realizado que teve resultado positivo, ou seja, vitória.

Com base nessa premissa foram desenvolvidas as estratégias A e B. A estratégia A procura na base histórica uma configuração exatamente igual à base corrente. A estratégia B é montada efetuando uma busca na base histórica de uma

configuração não necessariamente igual à configuração corrente, mas a configuração mais próxima à atual. Para essa métrica utilizam-se as regras de informações sobre a configuração do jogo e para cada posição igual a atribui-se um ponto a mais em uma nota de avaliação. A configuração com a maior pontuação é escolhida para servir de modelo e efetuar a escolha do lance, que no caso será idêntica ao lance efetuado no jogo histórico, no qual foi encontrada a configuração. Para exemplificar a pontuação, em um jogo de damas cada posição do tabuleiro preenchida com peça da mesma cor ou se em ambas não houver peças, atribui-se um ponto.

As estratégias C e D são elaboradas com base em um das configurações finais de vitória. Na elaboração da estratégia C escolhe-se, dentre as configurações finais encontradas, aquela que se encontrar mais próxima à configuração atual, ou seja, aquela que possui mais pontos, sendo que o critério de pontuação é o mesmo descrito para a estratégia B.

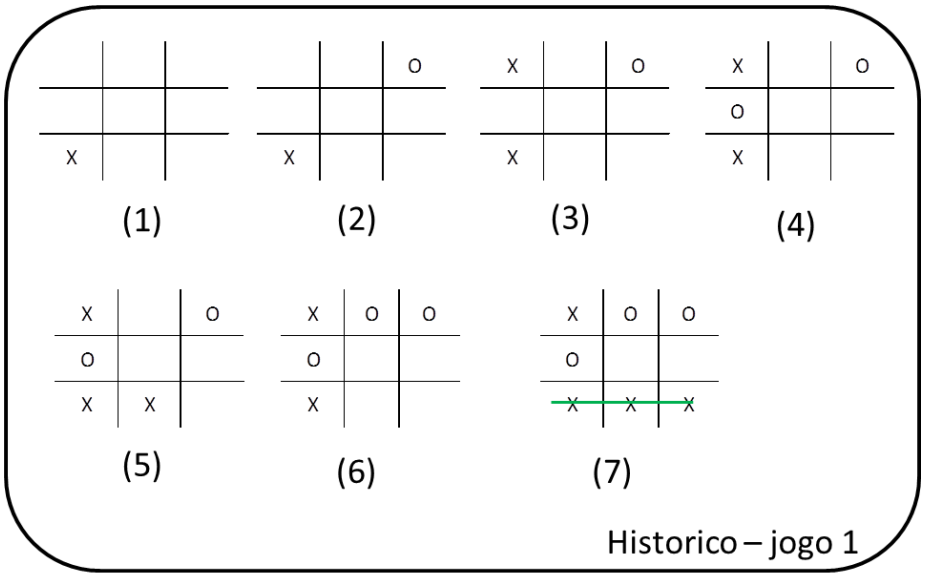
A estratégia D é efetuada com base na configuração final dos jogos efetuados também, porém nem toda configuração é considerada. Com base nas informações das regras que identificam o critério de término do jogo, efetua-se a identificação apenas das configurações relevantes para o critério. Por exemplo, no jogo da velha, apenas as posições que compõem a linha preenchida com o mesmo símbolo são informações relevantes na determinação do final da partida, as demais seis posições não interferem no critério.

Tabela 5.1- Resumo das estratégias.

Estratégia	*Descrição*
R	Escolha randômica entre as as possiveis jogadas
E	Entre os possíveis lances, verifica qual jogada leva a uma configuração que teve mais participações em jogos vitoriosos.
A	Escolhe, entre os possiveis lances, a que resultará em uma configuração idêntica a alguma configuração de um jogo vencedor do passado. Se não encontrar nenhuma opção, efetua a escolha de forma aleatória
B	Similar à estrategia A, porém caso não tenha uma configuração idêntica a um jogo vencedor do passado, escolhe-se o lance jogada que deixará a configuração mais próxima de uma configuração de um jogo vencedor.
C	Escolhe, entre as possiveis jogadas, a que resultará em uma configuração idêntica a alguma configuração final de um jogo vencedor do passado. Se não encontrar nenhuma configuração idêntica, escolhe a configuração mais próxima de uma configuração final, de um jogo vencedor.
D	Escolhe, entre os possiveis lances, a que resultará em uma configuração mais próxima de uma configuração final de um jogo vencedor, considerando-se apenas as posições consideradas na regra de término.

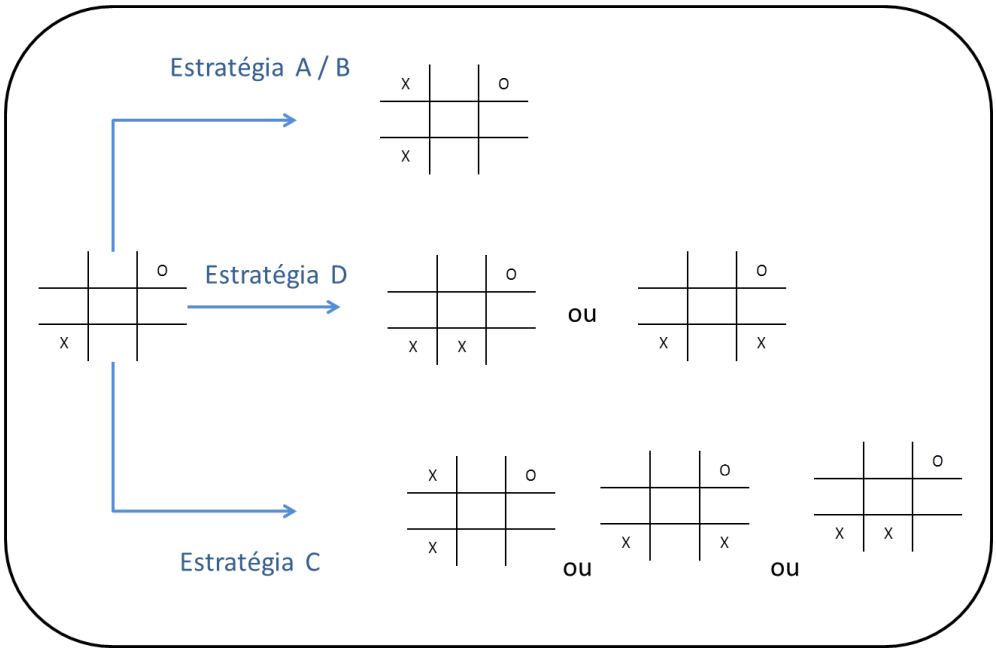
Fonte: Autor

Figura 5.1- Exemplo de jogo da velha.



Fonte: Autor

Figura 5.2- Alternativas em função das estratégias.



Fonte: Autor

A tabela 5.1 apresenta o resumo das estratégias e os seus respectivos critérios. Para ilustrar as diferenças das estratégias, consideramos a situação onde no histórico temos apenas um jogo, conforme ilustrado na figura 5.1. A figura 5.2 apresenta uma configuração, contendo uma posição preenchida com “O” e outra com “X”. O jogador, do papel “X”, teria apenas o lance na posição (1,1) se adotasse as estratégias “A” ou “B”, que considera a configuração (2) do histórico idêntica a sua posição, jogando assim de acordo com o lance (3) do jogo histórico. A estratégia “D” considera apenas as posições da linha horizontal que resultou no término do jogo, apresentando duas opções, correspondente à posição central e à posição (3,3). A estratégia C considera a configuração final do jogo histórico, podendo alterar a sua configuração para qualquer uma das configurações, onde há a marcação do “X” , podendo ser então igual às duas posições da estratégia D ou igual à posição da estratégia A e B.

5.2 Descrição dos experimentos

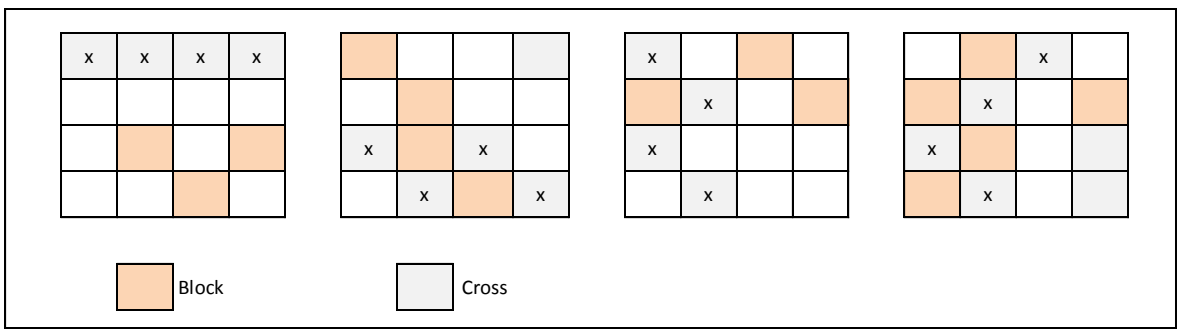
Efetuamos vários jogos entre duas instâncias da aplicação Adap-GGP, alternando a sua estratégia de jogo em cada situação e jogo, para análise da performance de cada estratégia.

Lembrando que a estratégia da aplicação não é especifica ao jogo em execução, visto que é elaborada com base no histórico dos jogos já efetuados e nas regras que definem o jogo. Os testes foram realizados utilizando-se duas instâncias de jogos, o jogo da velha e “block-cross”.

O “block-cross” é um jogo composto de uma matriz de quatro linhas e quatro colunas. Há dois papéis, o “cross” e o “block”. O “cross” tem que fetuar marcações que construam uma linha, horizontal ou vertical. Duas marcações consecutivas formam a mesma linha se a diferença, em valor absoluto, de um de seus índices é menor ou igual a um. Da mesma forma que ocorre no jogo da velha, uma mesma posição da matriz só pode ser preenchida por um dos dois participantes.

A figura 5.1 ilustra alguns exemplos de marcação que constituem linhas válidas. As células cinza representam os lances do “cross”, enquanto as vermelhas representam os lances do papel “block”. As células marcadas com “x” indicam as posições que formaram a linha, horizontal ou vertical.

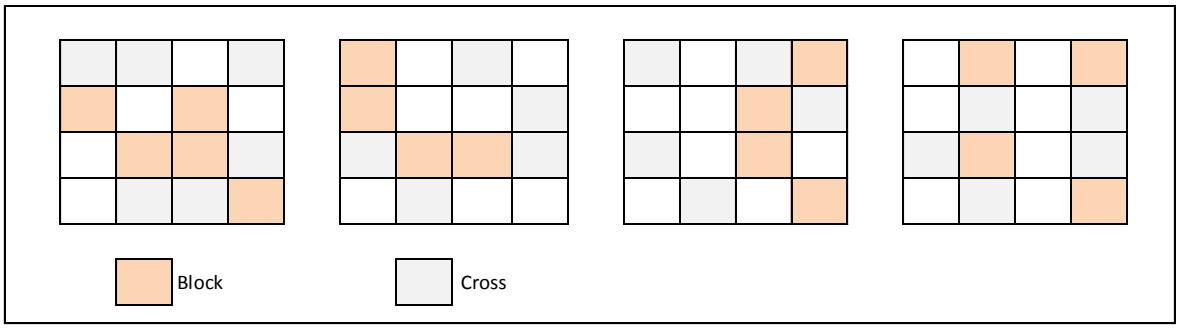
Figura 5.3- Exemplos de linhas válidas para o papel "cross".



Fonte: Autor

A figura 5.2 ilustra alguns exemplos de lances do papel “cross”, identificadas na cor cinza, que não tiveram êxito na formação de uma linha.

Figura 5.4- Exemplos de lances que não formam linhas.



Fonte: Autor

Premissas dos experimentos

Para que se tenha uma comparação dos resultados de cada uma das estratégias, um dos jogadores, sempre como mesmo papel em todos os jogos, foi mantido com a estratégia fixa, a aleatória, para todos os jogos. Assim, essa configuração é a base de comparação dos nossos experimentos.

5.3 Jogo da velha

As simulações utilizando as regras do jogo da velha seguiram os critérios:

a) o jogador, cujo papel do jogo é “O”, foi mantido fixo com a estratégia aleatória.

b) o jogador, cujo papel do jogo é “X”, foi executado com todas as estratégias, inclusive a aleatória, visto que as possibilidades de vitória dos papéis “O” e “X” são diferentes.

Tabela 5.2: Resultados em quantidades.

Jogo da velha			
Estratégia	Vitória	Empate	Derrota
R	575	157	268
A	690	115	195
B	754	99	147
C	688	112	200
D	874	42	84
E	598	160	242

Fonte: Autor

A tabela 5.2 apresenta os números absolutos, enquanto que a tabela 5.3 apresenta os percentuais, para cada uma das combinações da estratégia do papel “X”, enfrentando sempre o mesmo adversário, papel “O” com a estratégia aleatória.

Tabela 5.3- Resultados em valores percentuais.

Jogo da velha			
Estratégia	Vitória	Empate	Derrota
R	57,5	15,7	26,8
A	69,0	11,5	19,5
B	75,4	9,9	14,7
C	68,8	11,2	20,0
D	87,4	4,2	8,4
E	59,8	16,0	24,2

Fonte: Autor

Nota-se que a estratégia randômica, primeira linha do quadro, tem resultados diferentes conforme o seu papel, “O” ou “X”. Com os valores da amostra, temos que em um jogo aleatório a possibilidade de vitória do jogador cujo papel é “X” é de aproximadamente 57 %.

Para esse jogo, o par de estratégias “A” e “B”, cuja base é seguir o caminho através do qual um jogo vencedor teve, houve um aumento com relação à opção aleatória de 20% e 31%, porém a opção com melhor índice de mérito, 52%, foi a de estratégia “D”, que considera apenas as configurações relevantes do estado final.

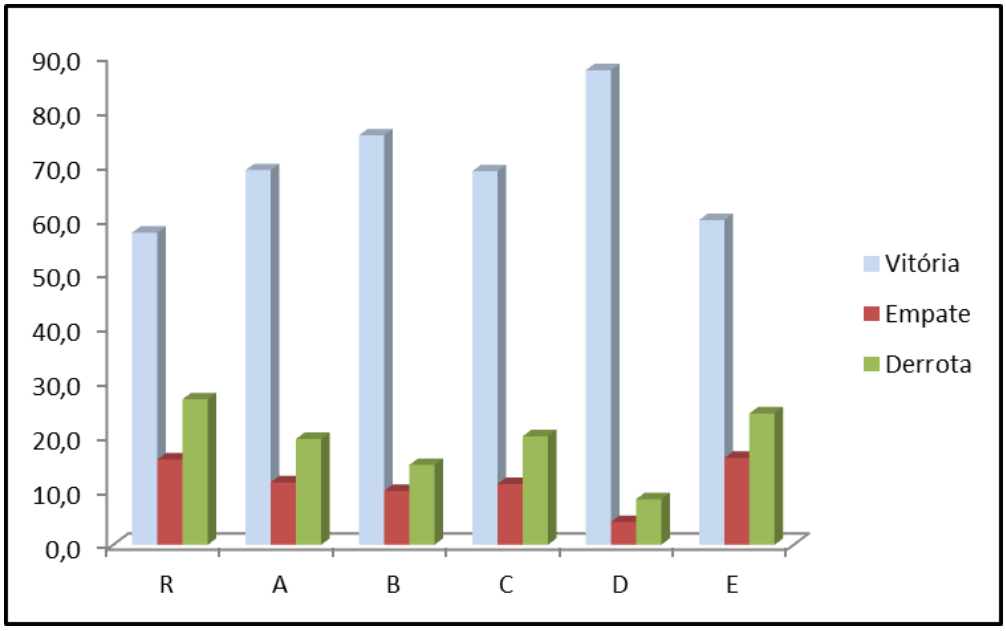
Tabela 5.4- Percentual de aumento de vitória, em relação à estratégia R.

Jogo da velha	
Estratégia	Vitória
A	20,0
B	31,1
C	19,7
D	52,0
E	4,0

Fonte: Autor

O gráfico da figura 5.3 apresenta a comparação entre as estratégias e seus respectivos resultados contra um mesmo adversário, de estratégia aleatória e papel “O”, conforme descrito no item “premissas dos jogos” da seção 5.1.

Figura 5.5- Valores percentuais das estratégias, para o papel "X", mantendo fixo as condições do papel “O”.



Fonte: Autor

5.4 Jogo “Cross-Block”

As simulações utilizando as regras do jogo “Cross-Block” seguiram as seguintes premissas:

a) um jogador, cujo papel do jogo é “Block”, foi mantido fixo com a estratégia aleatória.

b) o jogador, com o papel “Cross” foi executado com todas as estratégias, inclusive a aleatória, visto que as possibilidades dos jogadores dos papéis “Cross” e “Block” são distintas.

Tabela 5.5- Resultados em quantidades.

Cross-Block		
Estratégia	Vitória	Derrota
R	356	144
A	375	125
B	383	117
C	426	74
D	456	44
E	385	115

Fonte: Autor

A tabela 5.5 apresenta os números absolutos dos experimentos efetuados, conforme as premissas apresentadas nos itens a e b acima, para o jogo (“Cross-Block”), enquanto que a tabela 5.6 apresenta os resultados em valores percentuais, para cada uma das combinações da estratégia do papel “Cross”.

Tabela 5.6- Resultados em valores percentuais.

Cross-Block		
Estratégia	Vitória	Derrota
R	71,2	28,8
A	75,0	25,0
B	76,6	23,4
C	85,2	14,8
D	91,2	8,8
E	77,0	23,0

Fonte: Autor

Nota-se que a estratégia randômica, primeira linha da tabela 5.6, tem resultados diferentes conforme o seu papel, “Cross” ou “Bock”. Com os valores da amostra,

conclui-se que em um jogo aleatório a possibilidade de vitória do jogador cujo papel é “Cross” é de aproximadamente 71 %.

A diferença de possibilidade de vitória entre os papéis decorre do papel “Cross” efetuar o primeiro lance e seu objetivo ser diferente do papel “Block”.

Para esse jogo, o par de estratégias “A” e “B”, cuja base é seguir o caminho que um jogo vencedor teve, houve um aumento com relação à opção randômica, em torno de 5%, conforme os valores apresentados na tabela 5.7. Nessa mesma tabela, temos que a estratégia com maior performance, 28,1%, foi a de estratégia “D”, que considera apenas as configurações relevantes do estado final. A estratégia “C” teve um aumento de 19,7% em relação à opção randômica.

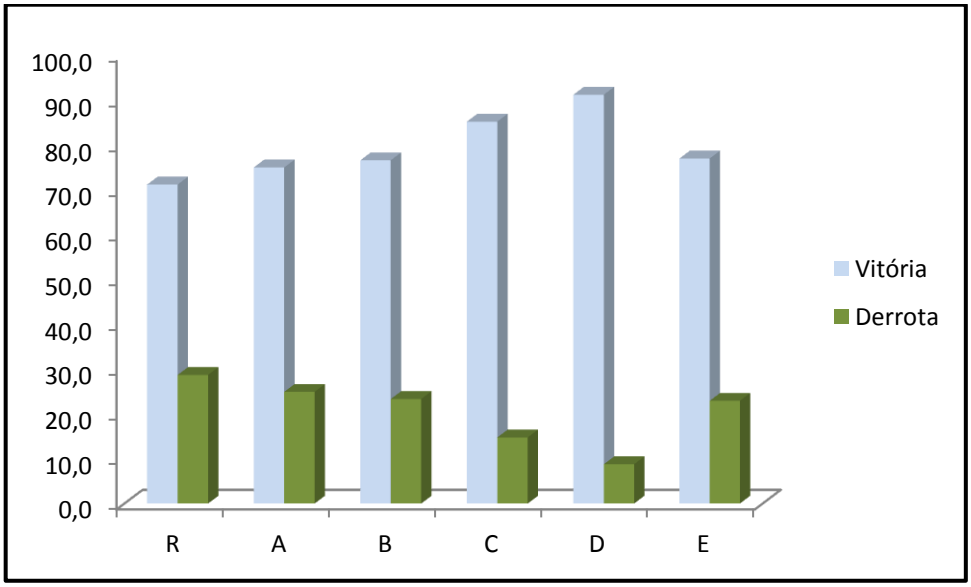
Tabela 5.7- Percentual de aumento de vitória, em relação à estratégia R.

Jogo Crosser-Block	
Estratégia	Vitória
A	5,3
B	7,6
C	19,7
D	28,1
E	8,1

Fonte: Autor

O gráfico da figura 5.4 apresenta a comparação entre as várias estratégias e seus respectivos resultados, para o papel “Cross”, contra um mesmo adversário, de estratégia randômica e papel “Block”.

Figura 5.6- Valores percentuais das estratégias, para o papel "Cross", mantendo fixas as condições do papel "Block".



Fonte: Autor

5.5 Conclusão dos experimentos

Considerando-se apenas os dois jogos simulados, a estratégia que apresentou melhores resultados foi a D.

A aplicação se mostrou aderente à arquitetura elaborada para dois dispositivos, mantendo assim separados os aspectos relacionados à execução do jogo e as questões relacionadas ao aprendizado da estratégia, baseada no histórico e nas informações contidas nas regras dos jogos.

Com a aplicação, tivemos resultados positivos na inferência de aprendizado a partir das informações contidas nas regras dos dispositivos, utilizadas em todas as estratégias.

6 CONCLUSÕES

Este capítulo descreve os resultados desta tese e a sua contribuição para a área de pesquisa. Finalizando, apresenta algumas sugestões para futuros trabalhos, que podem ser desenvolvidos a partir dos resultados alcançados por esta pesquisa.

6.1 Contribuições

Podemos separar em duas partes as contribuições dessa pesquisa. Em uma primeira parte, temos uma formulação teórica mais abrangente para utilização da adaptividade. Com essa formulação pode-se representar problemas complexos na modelagem de sistemas reativos, sistemas híbridos e sistemas concorrentes, nas mais diversas áreas como, biologia, medicina e ciências sociais.

Dentro da engenharia da computação, pode-se utilizar o modelo DAC na representação de aspectos da robótica, aprendizagem de máquina, processamento de linguagem natural e linguagem de programação.

Na linguagem de programação, pode-se aplicar o modelo considerando duas linguagens de programação voltadas à codificação de aplicações adaptativas, com uma camada de controle de supervisão, de modo que o comportamento de uma das linguagens possa afetar a outra adequadamente, mantendo sua integridade. Uma das linguagens nesse modelo poderia ser uma linguagem formal de especificação, enquanto a segunda poderia ser a linguagem de codificação.

Pode-se aplicar o modelo DAC também em duas gramáticas de linguagem natural, uma representando um núcleo comum e a outra representando as alterações regionais da língua, alterando algumas das suas regras. Assim podemos fatorar a diversidade regional de uma língua representada em uma gramática, que quando aplicada influenciará a gramática na forma padrão, gerando uma nova gramática, representando seu caráter regional.

Outra possibilidade é o uso do DAC na tradução de línguas naturais, representando em cada dispositivo um reconhecedor de uma língua natural, efetuando a tradução entre elas através de uma comunicação para cada par de reconhecedores.

Resumindo, o modelo DAC abre o leque de aplicações que podem empregar de modo formal a tecnologia adaptativa, utilizando vários tipos de modelos ou dispositivos.

A segunda parte se trata dos aspectos relacionados à aplicação desenvolvida. Foi mostrado que a adaptatividade pode ser aplicada na área de jogos GGP, uma linha de pesquisa promissora, trazendo resultados para outras áreas do conhecimento.

Com a arquitetura apresentada, podem-se desenvolver outras aplicações com o mesmo princípio. A representação de questionário de avaliação de um tópico de conhecimento, pode ser aplicado na mesma arquitetura, necessitando de algumas adequações. Um dos dispositivos representaria as alternativas respondidas por um aluno, enquanto o outro dispositivo monitoraria seu rendimento criando as alternativas para a próxima lição (próximo estado) de forma personalizada, com base no seu histórico. Resumindo, podemos facilmente usar a arquitetura para utilização de avaliação supervisionada de um determinado assunto.

A utilização de uma aplicação supervisionada pode ser empregada nas seguintes áreas:

- economia, em jogos de empresa
- biologia
- ciências cognitivas
- modelos de comportamento
- aprendizagem de máquina
- computação evolutiva
- agentes

- ensino por computador
- simulação
- realidade virtual
- projeto de veículos , como naves, etc
- treinamento de pilotos de avião
- aplicações médicas, como cirurgia a distância,
- robótica

6.2 Trabalhos futuros

Com o trabalho apresentado, surgem inúmeras oportunidades para trabalhos futuros, destacando-se entre tantas as seguintes:

- Ampliar as estratégias de aprendizado e a quantidade de jogos nas simulações, apresentando um resumo comparativo dos desempenhos das estratégias por categorias de jogos.

- Desenvolvimento de *framework* para criação de jogos GGP adaptativos.

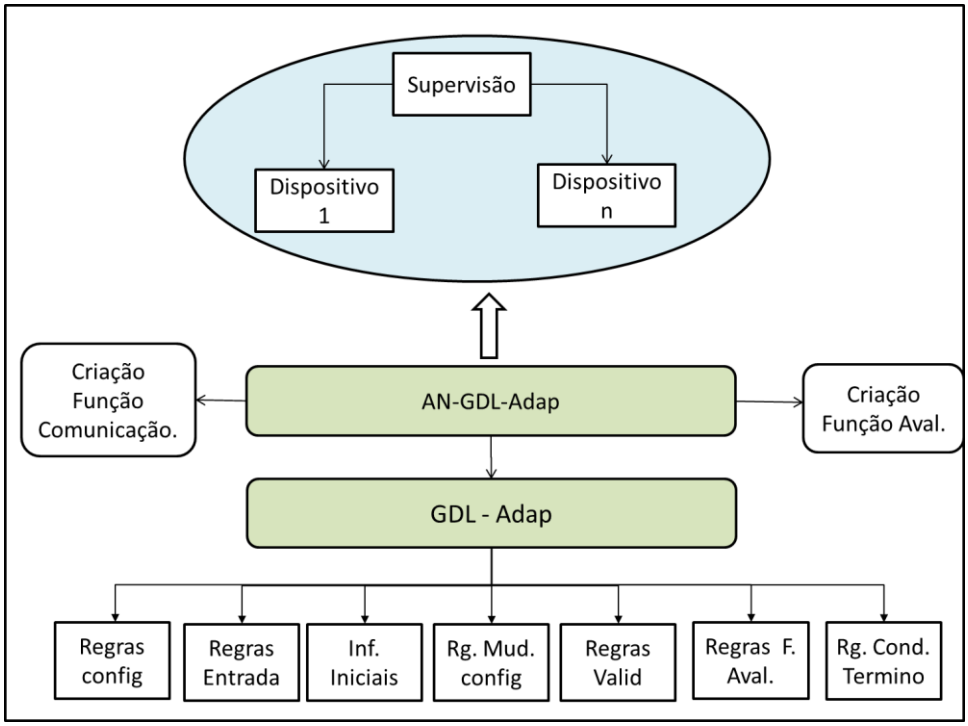
- Criação de um analisador sintático para verificação de especificação de jogos adaptativos, indicando se há solução (validador de especificação).

- Uma arquitetura geral para DAC, ilustrada nas figuras 6.1 e 6.2, para criação e simulação de dispositivos adaptativos cooperantes, através de vários módulos, como:

  - Linguagem GDL-adaptativa.

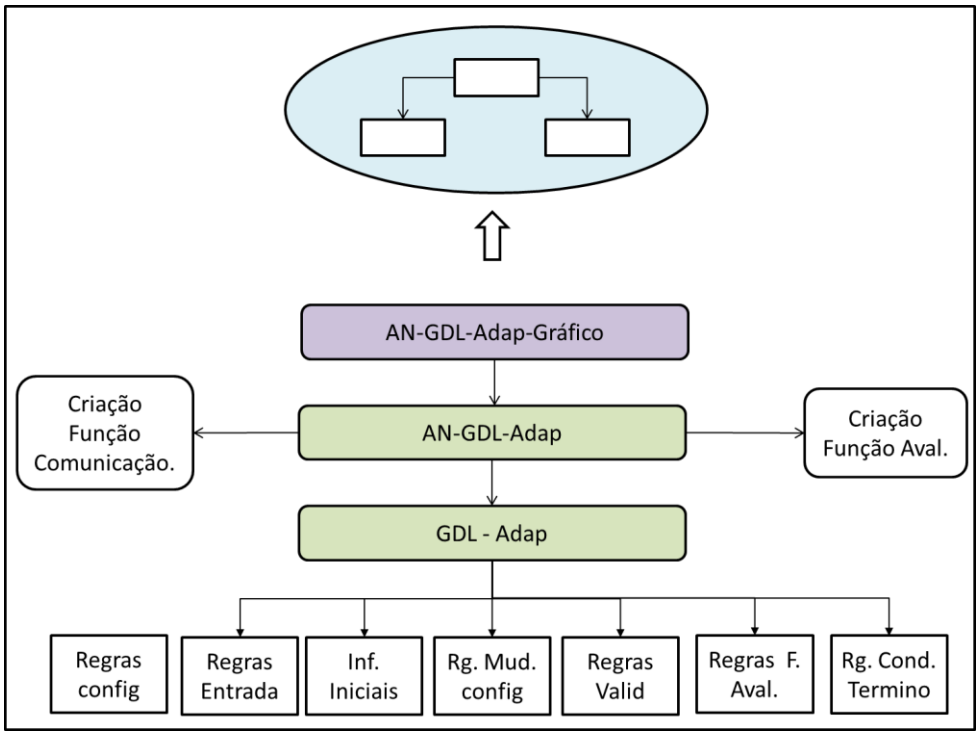
  - Linguagem de alto nível (AN-GDL-Adap), com geração do código na linguagem GDL-Adaptativo.

Figura 6.1– Arquitetura para criação e simulação de DAC



Fonte: Autor

Figura 6.2- Arquitetura com interface gráfica para ambiente de criação de DAC



Fonte: Autor

- Inclusão na linguagem funcionalidades para criação de mensagem de comunicação e função de avaliação para permitir a modelagem de dispositivos para supervisão ou de monitoramento de outros dispositivos.

- Criação de interface gráfica para a definição de todos os componentes da linguagem AN-GDL-Adap, gerando a versão AN-GDL-Adap-Gráfica.

REFERÊNCIAS

[1] ALMEIDA JUNIOR, J. R. **STAD: Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos**. Tese de Doutorado, Escola Politécnica, **Universidade de São Paulo**, São Paulo, 1995.

[2] ANTONY, P.; BALLING ,R., VLASSIS, N. From systems biology to systems biomedicine. **Current opinion in biotechnology** 23, no. 4, pp. 604-608, 2012.

[3] BAEK, S. J.; HAN, J. S.; CHUNG, K. Y. Dynamic reconfiguration based on goal-scenario by adaptation strategy. **Wireless personal communications** 73.2: 309-318, 2013.

[4] CASTRO, A. A.; NETO, J. J.; PISTORI, H. Deterministic Adaptive Finite Automata. **Revista IEEE América Latina**, v. 5, p. 515-521, 2007.

[5] CEREXHE, T.; SABUNCU, O.; THIELSCHER, M. Evaluating Answer Set Clause Learning for General Game Playing. In: **Logic Programming and Nonmonotonic Reasoning**. Springer Berlin Heidelberg, p. 219-232, 2013.

[6] CHATTERJEE, P.; CHAKRABORTY, S. Investigating the Effect of Normalization Norms in Flexible Manufacturing Sytem Se-lection Using Multi-Criteria Decision-Making Methods. **Journal of Engineering Science and Technology Review**, v. 7, n. 3, p. 141-150, 2014

[7] FREITAS, A. V.; NETO, J. J. Linguagens de Programação aderentes ao Paradigma Adaptativo. **Revista IEEE América Latina**, v. 5, p. 522-526, 2007.

[8] GENESERETH, M.; LOVE, N.; PELL, B. General game playing: Overview of the AAAI competition**. AI magazine**, v. 26, n. 2, p. 62, 2005.

[9] GENESERETH, M.; THIELSCHER, M. General Game Playing. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, v. 8, n. 2, p. 1-229, 2014.

[10] HEARST, M. A. Automatic acquisition of hyponyms from large text corpora. **Proceedings of the 14th conference on Computational linguistics**-Volume 2. Association for Computational Linguistics, 1992.

[11] HIRAKAWA, R.; SARAIVA A. M.; CUGNASCA C. E. Adaptive Automata Applied on Automation and Robotics. **Revista IEEE América Latina**, v. 5, p. 539-543, 2007.

[12] HOOD L.; GALAS D. J. Systems biology and emerging technologies will catalyze the transition from reactive medicine to predictive, personalized, preventive and participatory (P4) medicine. **Interdisciplinary Bio Central**, vol 1, no. 6, pp. 1-5, 2009.

[13] INGÓLFSDÓTTIR, A.; LARSEN, K. G.; SRBA, J. **Reactive systems: modelling, specification and verification**. Vol. 8, Cambridge, Cambridge University Press, 2007.

[14] KUHLMANN, G.; STONE, P. Automatic heuristic construction in a complete general game player. In: **AAAI**, p. 1457-1462, 2006.

[15] LEE Y., S.; CHO, S. B. A Hybrid System of Hierarchical Planning of Behaviour Selection Networks for Mobile Robot Control. **Int J Adv Robot Syst**, v. 11, p. 57, 2014.

[16] LEMOS, R.; GIESE, H.; MÜLLER, H. A.; SHAW, M.; ANDERSSON, J.; LITOIU, M.; WUTTKE, J. Software engineering for self-adaptive systems: A second research roadmap. In: **Software Engineering for Self-Adaptive Systems** II. Springer Berlin Heidelberg, p. 1-32, 2013.

[17] MAHEMUTI, P.; YANG, L.; LUO, L. L. Modeling, Analysis and Synthesis of Hybrid System: A Review. In: **Applied Mechanics and Materials**, p. 36-43, 2014.

[18] MORIN, E.; CHRISTIAN, J. Automatic acquisition and expansion of hypernym links. **Computers and the Humanities** 38.4: 363-396, 2004.

[19] NETO, J. J. Adaptative rule-driven devices – general formulation anda case study. In: **CIAA'2001 Sixth International Conference on Implementation and Application of Automata**, pages 234–250, Pretoria, South Africa, July 2001.

[20] NETO, J. J. Um levantamento da evolução da adaptatividade e da tecnologia adaptativa. **Revista IEEE América Latina**, v.5, n.7, p. 1548-0992, 2007.

[21] NETO, J.J. Um Levantamento da Pesquisa em Técnicas Adaptativas na EPUSP. **Revista de Sistemas e Computação** - RSC, v. 1, n. 1, 2011.

[22] NETO J. J.; ALMEIDA JUNIOR, J. R. Modeling Adaptive Reactive Systems. In: **IASTED International Conference Applied Modelling and Simulation,** Cairns. Applied Modelling and Simulation, p. 447-45, 1999

[23] NETO, J. J.; ALMEIDA JUNIOR, J. R.; SANTOS, J. M. N. Syncornized Statecharts for Reactive Systems. In: **International Conference on Applied Modelling and Simulation**, Honolulu, Hawaii. Proceedings of AMS'98 - IASTED,. p. 246-251, 1998.

[24] OKADA, R. S. **Tecnologia adaptativa aplicada a sistemas híbridos de apoio à decisão**. 2013. Dissertação (Mestrado em Sistemas Digitais) - Escola Politécnica, University of São Paulo, São Paulo, 2013.

[25] OTTE, C. An Independent General Game Player. In: **GI-Jahrestagung**, p. 134-136, 2013.

[26] PEDRAZZI, T. C.; TCHEMRA, A. H.; ROCHA, R. L. A. **Adaptive Decision Tables A Case Study of their Application to Decision-Taking Problems**. Springer Vienna, 2005.

[27] PELEGRINI, E. J.; NETO, J. J. A dynamically variable code execution model, based on adaptive automata. **Revista IEEE América Latina**, v. 6, p. 424-435, 2008.

[28] PILLAC, V.; GENDREAU, M.; GUÉRET, C.; MEDAGLIA; A. L. A review of dynamic vehicle routing problems. **European Journal of Operational Research**, v. 225, n. 1, p. 1-11, 2013.

[29] PISTORI, H. **Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.** Tese de Doutorado, EPUSP, Universidade de São Paulo, São Paulo, 2003

[30] PISTORI, H.; NETO, J. J.; PEREIRA M. C. Adaptive Non-Deterministic Decision Trees: General Formulation and Case Study. **INFOCOMP (UFLA)**, Lavras - MG, v. 5, p. 35-40, 2006.

[31] ROCHA, R. L. A. Tecnologia Adaptativa Aplicada ao Processamento Computacional de Língua Natural. **Revista IEEE América Latina**. Vol. 5, Num. 7, ISSN: 1548-0992, pp. 544-551, Novembro 2007.

[32] ROCHA, R. L. A.; NETO, J. J. Autômato Adaptativo, Limites e Complexidade em Comparação com Máquina de Turing. In: **Proceedings of the second Congress of Logic Applied to Technology – LAPTEC'2000**. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, pp. 33-48, 2001.

[33] RODRIGUES, E. S. C.; RODRIGUES, F. A; ROCHA, R. L. A.; CORREA, P. L. P. Adaptive Approach for a Maximum Entropy Algorithm in Ecological Niche Modeling. **Latin America Transactions, IEEE**, vol.9, no.3, pp.331,338, June, 2011.

[34] SABALIAUSKAS, J. A.; ROCHA, R. L. A. Project and Implementation for a Programming Language Suitable to Express Adaptive Algorithms. **Revista IEEE América Latina**, v. 9, p. 969-973, 2011.

[35] SALVADOR, R. B. S.; NETO, J. J. Proposal of a High-Level Language for Writing Self Modifying Programs. **Revista IEEE América Latina**, v. 9, p. 192-198, 2011.

[36] FREITAS, A. V.; NETO, J. J. Linguagens de Programação aderentes ao Paradigma Adaptativo. **Revista IEEE América Latina**, v. 5, p. 522-526, 2007.

[37] SANTOS, J. M. N. **Um formalismo adaptativo com mecanismo de sincronização para aplicações concorrentes**. Dissertação de Mestrado, Universidade de São Paulo, São Paulo, 1997.

[38] SAYAMA, H.; PESTOV, I.; SCHMIDT, J.; BUSH, B. J.; WONG, C.; YAMANOI, J.; GROSS, T. Modeling complex systems with adaptive networks. **Computers & Mathematics with Applications** 65.10: 1645-1664, 2013.

[39] SHANG, Y. An agent based model for opinion dynamics with random confidence threshold. **Communications in Nonlinear Science and Numerical Simulation** 19, no. 10, pp. 3766-3777, 2014.

[40] STANGE, R. L.; GIANNINI, T. C.; SANTANA, F. S.; NETO, J. J.; SARAIVA, A. M. Evaluation of Adaptive Genetic Algorithm to Environmental Modeling of Peponapis and Cucurbita. **Revista IEEE América Latina**, v. 9, p. 171-177, 2011.

[41] TAMURA, G.; VILLEGAS, N. M.; MÜLLER, H. A.; SOUSA, J. P.; BECKER, B.; KARSAI, G.; WONG, K. Towards practical runtime verification and validation of self-adaptive software systems. **Software Engineering for Self-Adaptive Systems II.** Springer Berlin Heidelberg, 108-132, 2013

[42] TCHEMRA, A. H. **Tabela de decisão adaptativa na tomada de decisão multicritério**. 2009. Tese (Doutorado em Sistemas Digitais) - Escola Politécnica, University of São Paulo, São Paulo, 2009.

[43] THYSSEN J.; BENJAMIN H. Behavioral specification of reactive systems using stream-based I/O tables. **Software & Systems Modeling** 12, no. 2, pp. 265-283, 2013

[44] VELARDI, P.; FARALLI, S.; NAVIGLI, R. OntoLearn Reloaded: A graph-based algorithm for taxonomy induction. **Computational Linguistics** 39.3: 665-707, 2013..

[45] VOGELSANG, A.; EDER, S.; FEILKAS, M.; RATIU, D. Functional Viewpoint. **Model-Based Engineering of Embedded Systems**, Springer Berlin Heidelberg, pp. 69-83, 2012.

[46] WOOLLEY, J. F.; STANICKA, J.; COTTER, T. G. Recent advances in reactive oxygen species measurement in biological systems. **Trends in biochemical sciences** 38, no 11, pp. 556-565, 2013.

APÊNDICE A

Apresenta os detalhes dos parâmetros das mensagens padrão do mecanismo de comunicação CC. O identificador *I-MGC* utilizado nos parâmetros das mensagens é um número inteiro e representa o mecanismo MGC. Os identificadores *I-AD-E* e *I-AD-D* são números inteiros e representam dispositivos adaptativos AD^r do DAC, respectivamente o emissor e o destinatário da mensagem.

- **Nome**: Inicial-Atribui-Identificador-Dispositivo

  Parâmetros de entrada: I-AD-D

  - ✓ I-AD-D - Número natural

  Retorno: Número de associação

  Destinatário: Dispositivo adaptativo AD^r

Através dessa mensagem atribui-se a um dispositivo AD^r, (para $1 \leq r \leq m$) tratado, um número de identificação único (parâmetro N1) dentro do grupo de dispositivos adaptativos cooperantes. Quando há sucesso na operação é retornado ao dispositivo o seu número de identificação, caso contrário o retorno é o valor Vazio.

- **Nome**: Inicial-Associação-Eventos-Entrada-Dispositivo

  Parâmetros de entrada: I-AD-D

  - ✓ I-AD-D - Número natural

  Retorno: Sucesso / Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Essa mensagem enviada pelo dispositivo I-AD-D, efetua uma associação dos seus elementos s_v^r do conjunto S^r ($s_v^r \in S^r$, v≥0), através de uma relação biunívoca, a um elemento de um conjunto de números naturais. Dessa forma, cada elemento do conjunto de eventos s_v^r pode ser referenciado através do número natural ao qual está associado, como exemplo o número 1, 7, etc., conforme ilustrado na figura 3.6.

Quando o dispositivo AD^r efetua essa associação para todos os eventos ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome:** Inicial-Associação-Eventos-Saida-Dispositivo

  Parâmetros de entrada: I-AD-D

  - ✓ I-AD-D - Número natural

  Retorno: Sucesso / Falha

  Emissor: MGC

  Destinatário: Dispositivo AD^r

Similar à mensagem *Inicia-Associação-Eventos-Entrada*, associa cada elemento z_r^I do conjunto NA^r ($z_v^r \in NA^r$, v≥0), através de uma relação biunívoca, a um elemento de um conjunto de números naturais. Dessa forma, cada elemento do conjunto de eventos de NA^r pode ser referenciado através do número natural ao qual está associado, como por exemplo 3, 5, etc.

Quando o dispositivo AD^r efetua essa associação para todos os eventos ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome:** Inicial-Associação-Configuração-Dispositivos

Parâmetros de entrada: I-AD-D

- ✓ I-AD-D - Número natural

Retorno: Sucesso / Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Similarmente, cada configuração $c^r_{kr,tk}$ ($c^r_{kr,tk} \in C^r_{kr,tk}$ kr≥ 0, tk≥0), do dispositivo r de DAC, é associada a um número natural através de uma relação biunívoca.

Quando o dispositivo AD^r efetua essa associação para todas as configurações ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome**: Inicial-Associação-Regras-Dispositivo

Parâmetros de entrada: I-AD-D

- ✓ I-AD-D - Número natural

Retorno: Sucesso / Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Cada regra $har^r \in HAR^r_{tk}$ (r>0, tk≥0) é associada a um número natural através de uma relação biunívoca.

Quando o dispositivo ADH^r efetua essa associação para todas as regras ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome**: Inicial-Associação-Ação adaptativa anterior-Dispositivo

  Parâmetros de entrada: I-AD-D

  - ✓ I-AD-D - Número natural

  Retorno: Sucesso / Falha

  Emissor: MGC

  Destinatário: Dispositivo AD^r

Cada ação adaptativa anterior do dispositivo é associada a um número natural através de uma relação biunívoca.

Quando o dispositivo AD^r efetua essa associação para todas as ações adaptativas anteriores ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome**: Inicial-Associação-Ação adaptativa posterior-Dispositivo

  Parâmetros de entrada: I-AD-D

  - ✓ I-AD-D - Número natural

  Retorno: Sucesso / Falha

  Emissor: MGC

Destinatário: Dispositivo AD^r

Cada ação adaptativa posterior do dispositivo é associada a um número natural através de uma relação biunívoca.

Quando o dispositivo AD^r efetua essa associação para todas as ações adaptativas posteriores ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome**: Inicial-Associação-Função Comunicação

Parâmetros de entrada: I-AD-D

- ✓ I-AD-D - Número natural

Retorno: Sucesso / Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Cada função de comunicação dos dispositivos é associada a um número natural através de uma relação biunívoca.

Quando o dispositivo efetua essa associação para todas as ações adaptativas posteriores ele retorna o valor "*Sucesso"*, caso contrário retorna como resposta a identificação de "F*alha*".

- **Nome**: Configura-Situação

Parâmetros de entrada: I-AD-D

- ✓ I-AD-D - Número natural

Retorno: Sucesso / Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Essa função atribui ao dispositivo a sua situação de ligado ou desligado conforme o parâmetro informado, respectivamente *ON* ou *OFF*. Caso ele consiga atribuir a situação informada ao dispositivo ele retorna como resposta o identificador *Sucesso*, caso contrário o identificador *Falha*.

- **Nome**: Informa-Situação

Parâmetros de entrada: I-AD-D

- ✓ I-AD-D - Número natural

Retorno: ON, OFF ou Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Essa mensagem retorna a configuração do dispositivo AD^r identificado por I-AD-D, ligado ou desligado (*ON* ou *OFF*, respectivamente).

Caso não se consiga efetuar a identificação de sua situação é retornado a identificação de “F*alha*”.

- **Nome**: Identificação-Regra

Parâmetros de entrada: I-AD-E, I-AD-D, N1, N2, N3, N4, N5, N6, N7, N8

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural
- ✓ N1- Número natural ou vazio
- ✓ N2 - Número natural ou vazio
- ✓ N3 - Número natural
- ✓ N4 - Número natural
- ✓ N5 - Número natural

- ✓ N6 - Número natural
- ✓ N7 - Número natural ou vazio
- ✓ N8 - Número natural ou vazio

Retorno: Número Natural ou vazio

Emissor: Dispositivo AD^E (identificado pelo número I-AD-E)

Receptor: Dispositivo AD^R (identificado pelo número I-AD-D)

Mensagem, enviada pelo dispositivo identificado por I-AD-E, que retorna o número natural associado à regra do dispositivo AD^r identificado pelo número I-AD-D, que satisfaz as condições informadas pelos parâmetros N1, N2, N3, N4, N5, N6, N7, N8.

Caso não se consiga efetuar a identificação da regra, o retorno da resposta é o identificador vazio.

- **Nome**: Criação-Regra

Parâmetros de entrada: I-AD-E, I-AD-D, N1, N2, N3, N4, N5, N6, N7, N8

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural
- ✓ N1- Número natural ou vazio
- ✓ N2 - Número natural ou vazio
- ✓ N3 - Número natural
- ✓ N4 - Número natural
- ✓ N5 - Número natural
- ✓ N6 - Número natural
- ✓ N7 - Número natural ou vazio
- ✓ N8 - Número natural ou vazio

Retorno: Número Natural ou vazio

Emissor: Dispositivo AD^e (identificado pelo número I-AD-E)

Receptor: Dispositivo AD^k (identificado pelo número I-AD-D)

Essa mensagem, enviada pelo dispositivo identificado por I-AD-E, cria uma regra no dispositivo AD^r (identificado por I-AD-D) com os respectivos parâmetros informados (N1, N2, N3, N4, N5, N6, N7, N8). Ao se criar a regra o dispositivo efetua a sua associação a um número natural ainda não utilizado por nenhuma regra, valor de retorno da mensagem.

Caso não se consiga criar a regra com os parâmetros informados o retorno da resposta é o identificador vazio.

- **Nome**: Exclusão-Regra

Parâmetros de entrada: I-AD-E, I-AD-D, N1

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural
- ✓ N1- Número natural ou vazio

Retorno: Sucesso ou Falha

Emissor: Dispositivo AD^E (identificado pelo número I-AD-E)

Receptor: Dispositivo AD^R (identificado pelo número I-AD-D)

Essa mensagem, enviada por I-AD-E, exclui a regra, do dispositivo identificado por I-AD-D, cujo número associado seja igual ao parâmetro informado (N1). Se o dispositivo, identificado por I-AD-D, conseguir excluir a regra retorna como resposta o identificador *Sucesso*, caso contrário o identificador *Falha*.

- **Nome**: Dispara-Evento

Parâmetros de entrada: I-AD-E, I-AD-D, N1

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural
- ✓ N1- Número natural ou vazio

Retorno: Sucesso ou Falha

Emissor: Dispositivo ADH^E (identificado pelo número I-AD)

Receptor: Dispositivo ADH^R (identificado pelo número I-AD-D)

O dispositivo, identificado por I-AD-E, solicita ao dispositivo identificado por I-AD-D disparar o evento, do seu conjunto S^k de eventos, cuja associação aos números inteiros seja igual ao parâmetro informado N1.

Se houver sucesso na operação é retornado o identificador *Sucesso*, caso contrário o identificador *Falha*.

- **Nome**: Novo-Estado

Parâmetros de entrada: I-AD-E, I-AD-D

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural

Retorno: Número inteiro N ou falha

Emissor: Dispositivo AD^E (identificado pelo número I-AD-E)

Receptor: Dispositivo AD^R (identificado pelo número I-AD-D)

Essa mensagem permite que o dispositivo, identificado por I-AD-E, solicite ao dispositivo identificado por I-AD-D que selecione uma nova configuração c_n^r de C^r, que ainda não esteja sendo utilizada em nenhuma regra. É similar à função geradora g_n das funções adaptativas. O retorno da mensagem é o número natural

associado à respectiva configuração. Caso ocorra algum erro, é retornado o identificador *Falha*.

- **Nome**: Identificação-Evento-Atual

  Parâmetros de entrada: I-AD-E, I-AD-D

  - ✓ I-AD-E - Número natural
  - ✓ I-AD-D - Número natural

  Retorno: Número inteiro N ou falha

  Emissor: Dispositivo AD^e (identificado pelo número I-AD-E)

  Receptor: Dispositivo AD^k (identificado pelo número I-AD-D)

Essa mensagem permite que o dispositivo, identificado por I-AD-E, solicite ao dispositivo identificado por I-AD-D o número associado ao seu atual evento.

- **Nome**: Identificação-Configuração-Atual

  Parâmetros de entrada: I-AD-E, I-AD-D

  - ✓ I-AD-E - Número natural
  - ✓ I-AD-D - Número natural

  Retorno: Número inteiro N ou falha

  Emissor: Dispositivo AD^e (identificado pelo número I-AD-E)

  Receptor: Dispositivo AD^k (identificado pelo número I-AD-D)

Essa mensagem permite que o dispositivo, identificado por I-AD-E, solicite ao dispositivo identificado por I-AD-D o número associado à sua configuração atual.

- **Nome:** Inicio-comunicação-entre-dispositivos

Parâmetros de entrada: I-AD-E, : I-AD-D, I-MGC

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural
- ✓ I-MGC - Número natural

Retorno: Sucesso / Falha

Emissor: Dispositivo AD^e (identificado pelo número I-AD-E)

Receptor: Dispositivo AD^r(identificado pelo número I-AD-D)

Gerenciador: I-MGC

- **Nome**: Fim-comunicação-entre-dispositivos

Parâmetros de entrada: I-AD-E, : I-AD-D, I-MGC

- ✓ I-AD-E - Número natural
- ✓ I-AD-D - Número natural
- ✓ I-MGC - Número natural

Retorno: Sucesso / Falha

Emissor: Dispositivo AD^e (identificado pelo número I-AD-E)

Receptor: Dispositivo AD^r(identificado pelo número I-AD-D)

Gerenciador: I-MGC

- **Nome**: Término

Parâmetros de entrada: I-AD-D

- ✓ I-AD-D - Número natural

Retorno: Sucesso / Falha

Emissor: MGC

Destinatário: Dispositivo AD^r

Essa mensagem solicite ao dispositivo identificado por I-AD-D desativar todas as associações vigentes.

ANEXO A

Exemplo da definição de um jogo, o jogo da velha, na linguagem GDL.

```
(role xplayer)
(role oplayer)

(index 1) (index 2) (index 3)
(<= (base (cell ?x ?y b)) (index ?x) (index ?y))
(<= (base (cell ?x ?y x)) (index ?x) (index ?y))
(<= (base (cell ?x ?y o)) (index ?x) (index ?y))
(<= (base (control ?p)) (role ?p))

(<= (input ?p (mark ?x ?y)) (index ?x) (index ?y) (role ?p))
(<= (input ?p noop) (role ?p))

(init (cell 1 1 b))
(init (cell 1 2 b))
(init (cell 1 3 b))
(init (cell 2 1 b))
(init (cell 2 2 b))
(init (cell 2 3 b))
(init (cell 3 1 b))
(init (cell 3 2 b))
(init (cell 3 3 b))
(init (control xplayer))

(<= (next (cell ?m ?n x)) (does xplayer (mark ?m ?n)) (true (cell ?m ?n b)))
(<= (next (cell ?m ?n o)) (does oplayer (mark ?m ?n)) (true (cell ?m ?n b)))
(<= (next (cell ?m ?n ?w)) (true (cell ?m ?n ?w)) (distinct ?w b))
```

(<= (next (cell ?m ?n b)) (does ?w (mark ?j ?k)) (true (cell ?m ?n b)) (or (distinct ?m ?j) (distinct ?n ?k)))
(<= (next (control xplayer)) (true (control oplayer)))
(<= (next (control oplayer)) (true (control xplayer)))

(<= (row ?m ?x) (true (cell ?m 1 ?x)) (true (cell ?m 2 ?x)) (true (cell ?m 3 ?x)))
(<= (column ?m ?x) (true (cell 1 ?m ?x)) (true (cell 2 ?m ?x)) (true (cell 3 ?m ?x)))
(<= (diagonal ?x) (true (cell 1 3 ?x)) (true (cell 2 2 ?x)) (true (cell 3 1 ?x)))
(<= (diagonal ?x) (true (cell 1 1 ?x)) (true (cell 2 2 ?x)) (true (cell 3 3 ?x)))

(<= (line ?x) (column ?m ?x))
(<= (line ?x) (row ?m ?x))
(<= (line ?x) (diagonal ?x))

(<= open (true (cell ?m ?n b)))

(<= (legal ?w (mark ?x ?y)) (true (cell ?x ?y b)) (true (control ?w)))
(<= (legal xplayer noop) (true (control oplayer)))
(<= (legal oplayer noop) (true (control xplayer)))

(<= (goal xplayer 100) (line x))
(<= (goal xplayer 50) (not (line x)) (not (line o)) (not open))
(<= (goal xplayer 0) (line o))
(<= (goal oplayer 100) (line o))
(<= (goal oplayer 50) (not (line x)) (not (line o)) (not open))
(<= (goal oplayer 0) (line x))

(<= terminal (line x))
(<= terminal (line o))
(<= terminal (not open))