Identificação de Sistemas Dinâmicos Finitos
e aplicações na modelagem de redes gênicas

Nestor Walter Trepode

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO

PARA OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: **Ciência da Computação**
Orientador: **Prof. Dr. Junior Barrera**

O autor recebeu apoio financeiro da **FAPESP**,
Fundação de Amparo à Pesquisa do Estado de São Paulo
– Processo Nº 00/10684-4 –

São Paulo, fevereiro de 2002

Identificação de Sistemas Dinâmicos Finitos

e aplicações na modelagem de redes gênicas

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e apresentada por **Nestor Walter Trepode**
e aprovada pela Comissão Julgadora.

São Paulo, fevereiro de 2002.

Banca Examinadora:

- **Prof. Dr. Junior Barrera** (orientador) (IME-USP)
- **Profa. Dra. Nina Sumiko Tomita Hirata** (IME-USP)
- **Prof. Dr. Paulo Agozzini Martin** (IME-USP)

aos meus pais,

Ana María e Humberto,

a minha irmã Sonia

e ao meu sobrinho Sebastian

Agradecimentos

Gostaria de agradecer a todas as pessoas que nestes anos no IME-USP me apoiaram e ajudaram, tornando mais fácil e agradável o desenvolvimento de minhas tarefas, isto é, aos meus companheiros de mestrado e do laboratório de processamento de imagens, professores e amigos.

Em particular,

agradeço especialmente ao meu orientador Junior Barrera, pela sua força motivadora, seu entusiasmo e atitude sempre positiva, pelo seu apoio e por tudo o que dele aprendi e espero continuar aprendendo;

tenho muito que agradecer a Nina e Roberto Hirata pela sua inestimável colaboração durante a realização do trabalho;

agradeço ao amigo Marcel Brun pela sua ajuda e "impulso inicial" no começo do meu mestrado;

agradeço a colaboração do professor Hugo Aguirre Armelin, que muito interagiu com meu orientador, e ao projeto CAGE, "Cooperation for Analysis of Gene Expression", que o patrocinou.

muito tenho que agradecer ao professor Marco Dimas Gubitoso pela sua paciência, boa disposição e colaboração durante o uso do simulador desenvolvido por ele;

agradeço também aos professores Routo Terada e Roberto Marcondes César Júnior pela sua ajuda na primeira parte do meu mestrado;

agradeço aos funcionários e funcionárias da biblioteca, da secretaria do departamento de Ciência da Computação e da secretaria de pós-graduação, que tão bem me trataram e ajudaram tanto na superação de inconvenientes e dificuldades, como nas tarefas do dia a dia;

agradeço à FAPESP, Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo apoio financeiro recebido durante este período;

e agradeço muito especialmente à minha família pelo apoio incondicional, carinho e incentivo permanente para o meu desenvolvimento pessoal.

A todos vocês, muito obrigado!

Resumo

O objetivo do presente trabalho é a *identificação de sistemas dinâmicos finitos*, nos quais o tempo é considerado discreto e à escala de valores discreta e finita. Estes sistemas modelam adequadamente a evolução temporal dos graus de ativação dos genes numa rede de expressão gênica (REG). Os *genes* que integram uma rede deste tipo são segmentos de DNA que codificam proteínas específicas. A *expressão gênica* consiste de dois passos: i) *transcrição* da informação codificada no *DNA* em moléculas de RNA e ii) *tradução* da informação codificada no *mRNA* (um dos tipos de RNA sintetizados no passo anterior) em uma seqüência definida de amino ácidos de uma *proteína*. As proteínas produzidas, como conseqüência da expressão gênica, formam complexos multiproteicos inter-atuantes os quais enviam sinais ao núcleo da célula para mudar os padrões da expressão gênica. Assim, os genes formam uma *rede*, que é chamada de *expressão gênica*, na qual o grau de ativação (ou nível de expressão) de cada gene depende dos níveis de expressão dele mesmo e de outros genes em instantes anteriores, e de estímulos externos.

O *estado* de um sistema dinâmico finito num instante de tempo é definido pelo valor de um vetor de variáveis chamadas de *variáveis de estado*, no caso de uma REG, o nível de expressão de cada gene. O valor de uma variável de estado num instante de tempo depende dos estados anteriores e de estímulos externos em instantes anteriores. As transições de estado são definidas pela *função de transição* do sistema, que é na verdade um vetor de funções cujas componentes determinam as transições de cada variável de estado.

Foram realizadas a *modelagem* e *simulação* do *ciclo celular* hipotético, a partir de fatos conhecidos e conjecturas sobre a dinâmica desse fenômeno.

Para testar técnicas de identificação de sistemas, foi simulado também um sistema mais simples, sem vinculo com fenômenos biológicos. A partir de *amostras* incompletas das transições de estado destas simulações, foi realizada a *identificação do sistema dinâmico* por técnicas de aprendizado computacional. Tal identificação consiste em achar as componentes da função de transição. Também foi efetuada a identificação com a aplicação de *restrição de envelope dinâmico* (limite inferior e superior das dinâmicas). Finalmente, ambos os sistemas identificados (com e sem restrição de envelope) foram simulados e comparados com as respectivas simulações do sistema ideal, medindo-se os *erros de estimação*.

Também foi desenvolvida uma técnica de geração automática de árvores de classificação por multirresolução, a qual poderá vir a ser aplicada no futuro à identificação de sistemas dinâmicos finitos.

Índice

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

Os cromossomos do núcleo de uma célula são estruturas auto-replicáveis que contém o *DNA –ácido desoxi-ribonucléico–*, molécula que codifica a informação genética. O DNA possui na sua seqüência de nucleotídeos –sub-unidade de DNA ou RNA– um arranjo linear de genes. Eles estão formados por sub-seqüências ordenadas de nucleotídeos e constituem a unidade física e funcional fundamental da hereditariedade (Figuras 1.1 e 1.2). Os genes possuem funções específicas e, para realizá-las, cada gene codifica um produto funcional determinado, i.e., uma proteína ou uma molécula de RNA –ácido ribonucléico–, que é produzido quando ele é expresso. A atividade da célula é determinada por quais genes estão sendo expressos ou não. As *proteínas* são necessárias para a estrutura, função e regulação das células do corpo, tecidos e órgãos, e cada proteína tem funções específicas. Hormônios, enzimas e anticorpos são exemplos de proteínas.

A *expressão gênica* é o processo que faz que a informação codificada nos genes seja convertida nas estruturas que estão presentes e atuam dentro da célula. Ela ocorre em

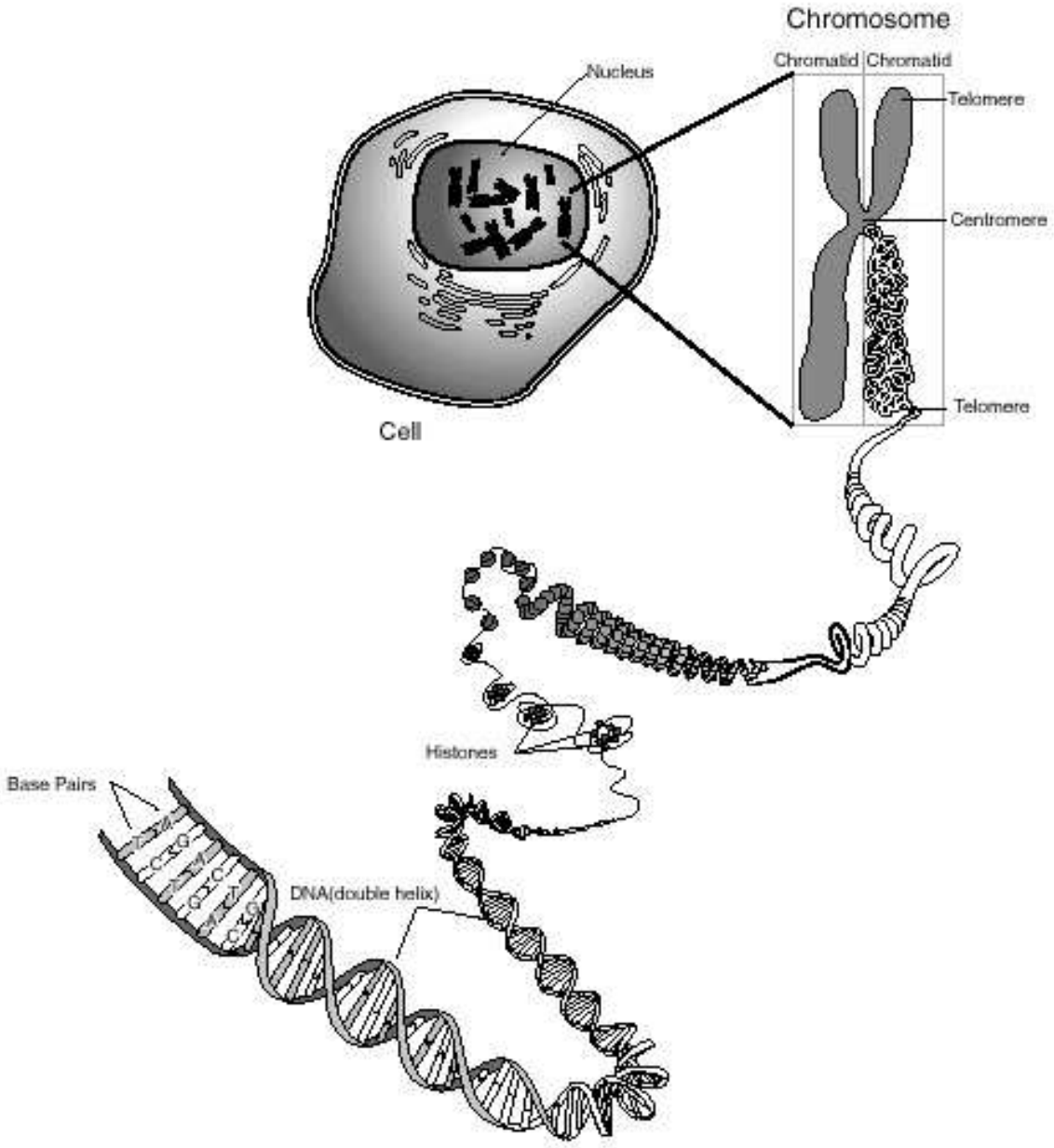


Figura 1.1: Cromossomo *(fonte: NHGRI, National Human Genome Research Institute).*

dois passos (Figura 1.3): a) *transcrição* da informação codificada em um gene, mediante a síntese de moléculas de RNA, i.e., cópias dele que são enviadas para fora do núcleo

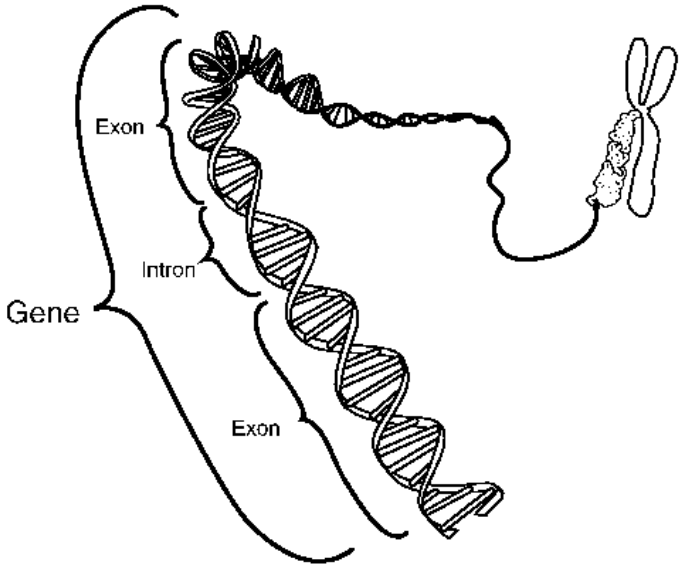


Figura 1.2: Gene *(fonte: NHGRI, National Human Genome Research Institute).*

da célula, e b) *tradução* da informação codificada nos nucleotídeos do mRNA (RNA mensageiro: um dos tipos de RNA produzidos na transcrição), mediante a síntese de uma proteína, a qual está formada por uma seqüência perfeitamente definida de amino ácidos.

O processo da expressão gênica é regulado por um complexo mecanismo de sinais celulares (Figura 1.4). As proteínas, produzidas como conseqüência da expressão dos genes, formam complexos multiproteicos inter-atuantes, que interagem com sinais externos e, ao mesmo tempo, enviam sinais de realimentação ao núcleo da célula, os quais mudam os padrões de expressão gênica. Os genes e os produtos da expressão gênica formam assim uma rede de transmissão de sinais que regula o funcionamento da célula. Esta rede é chamada de *rede de expressão gênica* (ou de regulação gênica), onde o grau de ativação (ou nível de expressão) de cada gene depende dos níveis de expressão dele mesmo e de outros genes em instantes anteriores, como também de estímulos externos.

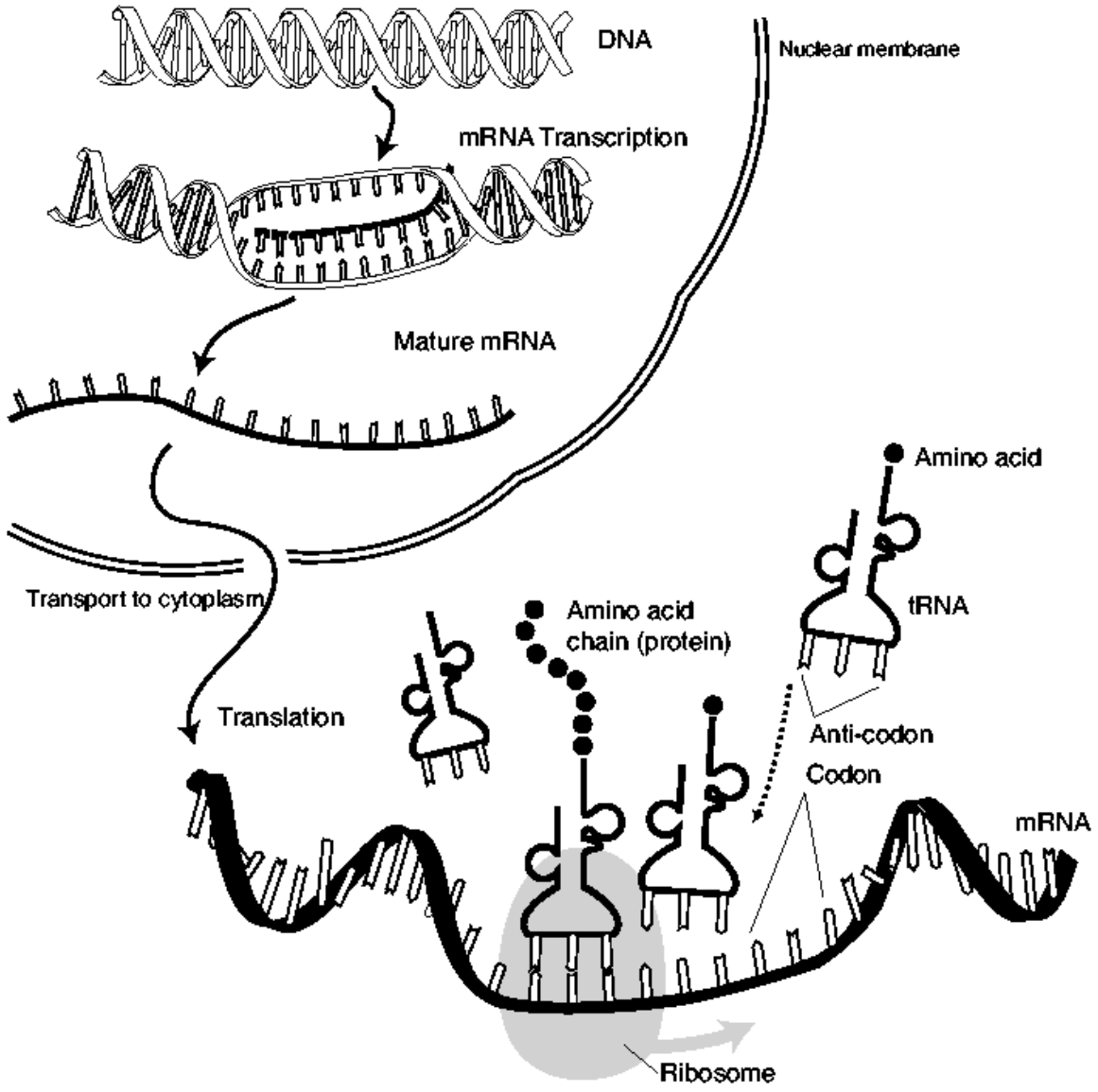


Figura 1.3: Expressão gênica *(fonte: NHGRI, National Human Genome Research Institute).*

Os dados experimentais que se dispõe na atualidade para a medição da expressão gênica são de dois tipos: *níveis de mRNA* e *níveis de proteínas* [3].

Os níveis de proteína constituem uma parte importante do funcionamento interno da célula. Muitas interações na célula ocorrem completamente no nível das proteínas e a

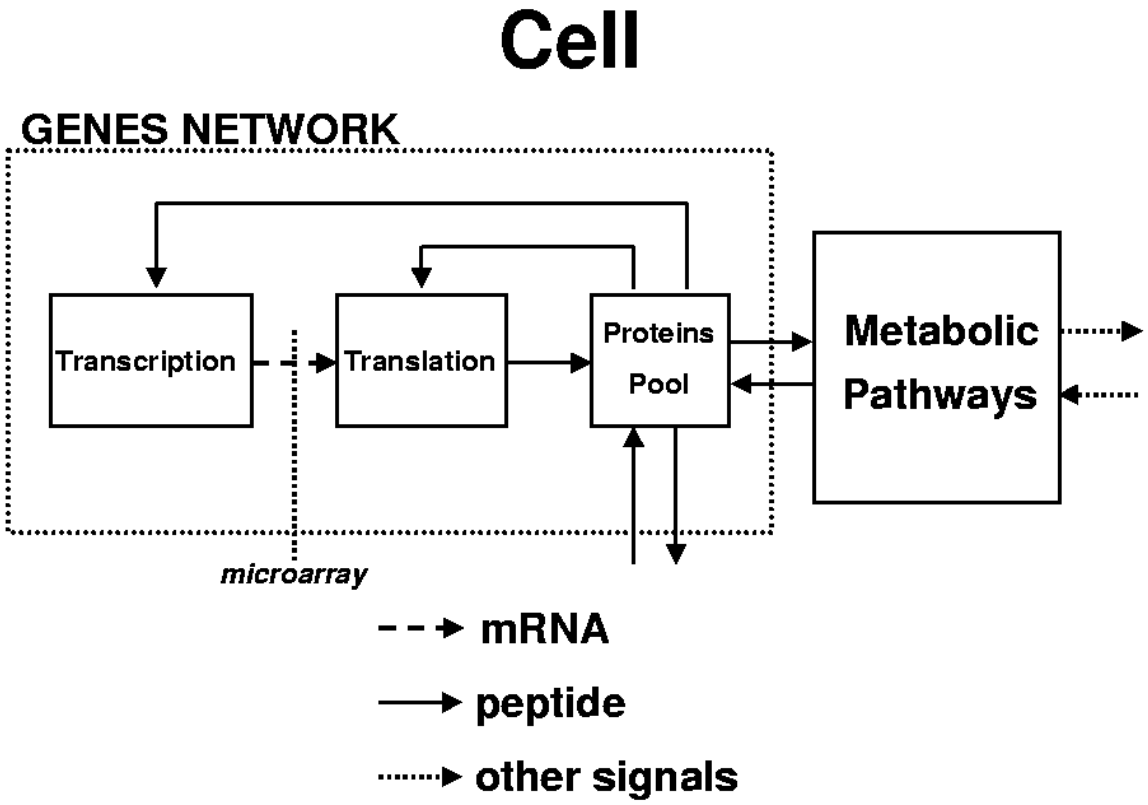


Figura 1.4: Rede de expressão gênica.

presença ou ausência de determinadas proteínas determinam a expressão ou não dos genes no DNA. Mas, na prática, os níveis de proteína são muito mais difíceis de quantificar do que os níveis de mRNA.

Podemos medir o nível de expressão de cada gene medindo quantas cópias de mRNA estão presentes na célula. A medição dos níveis de mRNA pode ser feita usando:

- Microarrays de cDNA
- Chips de oligonucleotídeos
- RT-PCR ("Reverse Transcriptase Polymerase Chain Reaction")

- Análise Serial da Expressão Gênica (SAGE: "Serial Analysis of Gene Expression")

O método mais freqüentemente utilizado é o dos *microarrays de cDNA*. Eles são adequados para a análise de até 10.000 clones de cDNA –"copy DNA"–. São formados por uma lâmina de vidro sobre a qual se depositam, por impressão robótica de alta velocidade, clones de cDNA num arranjo de pontos microscópicos –ou *spots*– (Figura 1.5). Cada spot contém muitas cópias de uma mesma seqüência de cDNA perfeitamente conhecida, que representa um gene específico e servirá para medir seu nível de expressão.

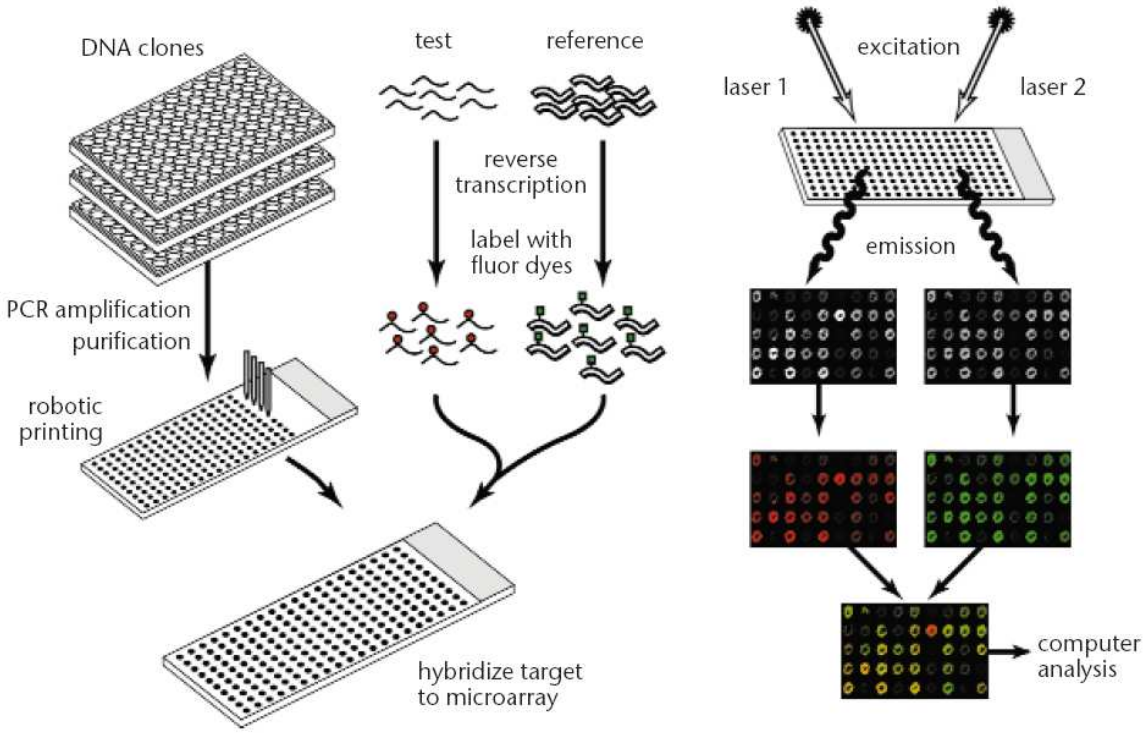


Figura 1.5: Tecnologia de Microarray [1].

Uma amostra do mRNA que se deseja analisar é extraída e retrotranscrita[1] para cDNA. Durante a retrotranscrição, marcam-se os cDNAs que estão sendo sintetizados com um

[1]Transcrição reversa: o processo de copiar em DNA a informação encontrada no RNA

fluoróforo[2]. Para evitar problemas de saturação, e por não podermos saber com exatidão quantas cópias de cada cDNA estão presentes em cada spot, usa-se uma referência contra a qual se comparam os níveis de expressão de cada gene. Assim, paralelamente à extração e tratamento do mRNA da amostra, se extrai e se trata da mesma forma o mRNA de uma amostra de referência, só que neste caso se usa um marcador fluorescente distinto, normalmente de outra cor. Os marcadores mais usados são Cy3 –verde– e Cy5 –vermelho–.

Misturam-se em quantidades iguais os dois cDNAs marcados e colocam-se em contato com o cDNA do array. Os cDNAs da mistura se unem ao spot que contenha a base complementar, este processo é chamado de *hibridização*. O processo é competitivo: a cada spot se unem mais ou menos cDNAs marcados de uma amostra ou de outra em função da abundância relativa deles na mistura inicial. Depois da hibridização, se excita o microarray com luz laser de dois tipos. Um digitalizador –"scanner"– produz imagens que medem os níveis dos dois tipos de fluorescência nos spots da lâmina. A combinação destas imagens dá origem à imagem que será usada na análise. Nela, cada spot aparece de uma cor ou de outra, dependendo se o gene representado se expressa mais em uma amostra ou na outra. A razão das intensidades de fluorescência em cada spot indica a abundância relativa da correspondente seqüência de DNA nas duas amostras de ácidos nucleicos.

Para ter uma idéia do que esta expressão diferencial é capaz de medir, podemos dizer, por exemplo, que os mRNAs da amostra e da referência podem pertencer a uma célula doente e a uma sadia, a células com dois tipos diferentes de câncer, a dois instantes de tempo diferentes, etc..

Atualmente, os níveis de mRNA são usados na maior parte dos estudos que visam

[2]Grupo de átomos necessários para a produção de fluorescência numa molécula.

inferir mecanismos de expressão a partir de medições experimentais na célula, principalmente, devido às tecnologias de expressão gênica de grande escala fazerem com que eles sejam as variáveis importantes mais fáceis de medir [4, 5, 6, 7].

As atuais tecnologias possibilitam a obtenção de visões globais dos padrões de atividade biomolecular que estão por trás de processos biológicos complexos, como a divisão e diferenciação celular. Esta disponibilidade de dados permite pensar em desafios tais como: a) inferência de relações funcionais importantes a partir destes dados, b) visualização destes resultados usando ferramentas gráficas e c) projeto de experimentos e estratégias de análise computacional, a fim de desvendar o significado dos dados de que se dispõe.

Os *sistemas dinâmicos finitos* ou FDS ("Finite Dinamical Systems") são adequados para modelar uma rede de expressão gênica. Neste modelo o nível de expressão de cada gene é representado pelo valor de uma *variável de estado*. O *estado* de um FDS é definido pelo valor num dado instante de todas as variáveis de estado. O valor de uma variável de estado num instante de tempo depende dos estados anteriores e de estímulos externos anteriores, através de uma função que representa os mecanismos de regulação gênica descritos anteriormente. Cada uma destas funções é uma componente de um vetor de funções chamado de *função de transição*, a qual define a transição de um estado do FDS para o seguinte.

A *identificação* de um sistema dinâmico finito consiste em encontrar as componentes da função de transição a partir de amostras dos estados do sistema. Enquanto a medição de muitas variáveis pode permitir a obtenção de um modelo mais exato, a complexidade do modelo (e a dificuldade para achá-lo) aumenta exponencialmente com o número de variáveis. A teoria de Aprendizado Computacional (que se aplica para a identificação)

explica que quanto maior for o número de variáveis necessárias mais difícil é a tarefa da modelagem, porque o tamanho do espaço de busca (número de modelos possíveis) aumenta exponencialmente com o número de parâmetros do modelo. Com a diminuição do tamanho do espaço dos modelos plausíveis, mediante a aplicação de restrições adicionais, procura-se simplificar consideravelmente a busca do melhor modelo. A restrição do modelo pelo uso de informação prévia acerca do que é biologicamente conhecido ou possível é, provavelmente, uma das mais importantes estratégias de que se dispõe para lidar com o problema de dimensionalidade, que surge como conseqüência do grande número de variáveis envolvidas e das múltiplas dependências funcionais possíveis entre estas.

1.2 Metodologia Adotada

A metodologia adotada é orientada ao desenvolvimento de técnicas de identificação de sistemas dinâmicos finitos. Tal identificação consiste na determinação das funções componentes da função de transição do FDS, a partir de amostras de seus estados.

A teoria de Aprendizado Computacional ("Machine Learning") [8] estabelece uma fundamentação teórica e metodológica para o aprendizado de funções partindo de exemplos que as amostram. Dado um conjunto de entradas e as correspondentes saídas de uma função, usado como exemplos de treinamento, um algoritmo de aprendizado produz uma estimativa da função ψ (Figura 1.6). A Figura 1.7 mostra com mais detalhe este processo. No passo 1, coleta de amostras, é criada uma tabela de freqüências de cada saída para cada entrada presente nas amostras de treinamento. No passo 2, para cada entrada encontrada nas amostras se decide pela saída mais freqüente gerando-se assim uma tabela de decisão. A partir desta tabela, no passo 3, a função ψ é estimada mediante

os processos de generalização e minimização.

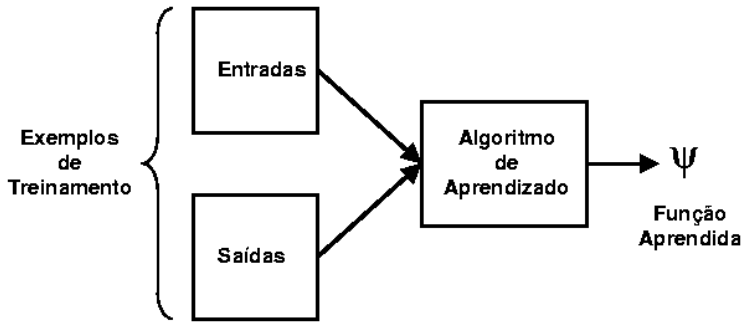


Figura 1.6: Aprendizado.

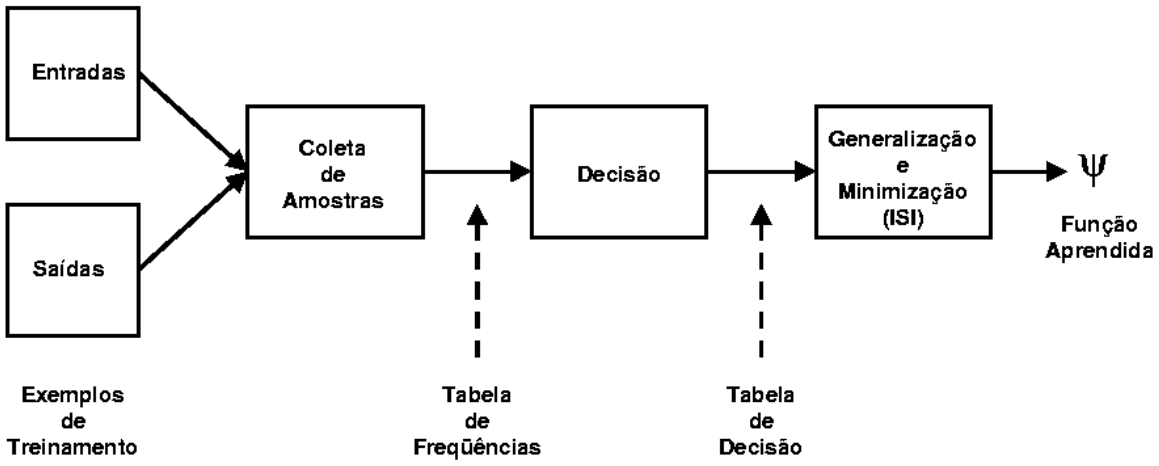


Figura 1.7: Etapas do Aprendizado Computacional.

O uso de algoritmos de *minimização* de funções permite encontrar expressões equivalentes mais simples, facilitando a manipulação das mesmas e reduzindo a quantidade de memória e o tempo de processamento requeridos [9, 10, 11]. O algoritmo ISI [12] ("Incremental Splitting of Intervals"), além de cumprir satisfatoriamente com esta função, também realiza a tarefa de *generalização*, que consiste em atribuir valores de saída para

aquelas combinações das variáveis de entrada para as quais não se dispõe de exemplos nos dados de treinamento.

Com referência aos erros cometidos, o modelo de aprendizado PAC ("Probably Approximately Correct") estipula que o erro cometido no aprendizado pode ser reduzido a um valor tão pequeno como se deseje, com uma probabilidade tão próxima de 1 (um) quanto se deseje, se a amostra de treinamento contém um número suficientemente grande de exemplos [13, 14, 15, 16]

Para efetuar a identificação por aprendizado das funções de transição deve escolher-se uma representação matemática que seja adequada. Nós utilizamos os *Sistemas Dinâmicos Finitos* ou FDS ("Finite Dinamical Systems") que podem manipular convenientemente os dados reais provenientes de medições experimentais (por exemplo, imagens de microarray) constituídos por valores discretos que representam os níveis de expressão dos genes em diferentes instantes de tempo. A evolução temporal de um FDS está dada por transições de estado. O valor do próximo estado de cada variável de estado é uma função, que é uma componente da *função de transição*, do valor das entradas e das variáveis de estado nos instantes anteriores. A cada transição de estados corresponde também uma saída, em que o valor de cada variável de saída é também uma função das variáveis de entrada e de estado. Num sistema *livre de entradas* o estado atual depende só dos estados anteriores. Se a função de transição está definida sobre um reticulado finito, o sistema é um *Sistema Dinâmico Finito de Reticulados* ou FLDS ("Finite Lattice Dinamical System"). Nesta representação, as componentes da função de transição são *operadores de reticulados*, representados pelo conjunto de suas *bases*, que são coleções minimais de intervalos maximais [17]. A estrutura de reticulados de um FLDS é importante para representar as componentes da função de transição na forma gerada pelo algoritmo ISI (coleção minimal

de intervalos maximais) usado no processo de aprendizado na generalização e minimização.

As técnicas de identificação foram estudadas num caso simples, partindo de um sistema dinâmico finito de reticulados. Tal sistema foi simulado e, usando amostras incompletas das transições de estado, foi efetuada a *identificação* do sistema por aprendizado computacional. A identificação do FLDS consiste na aplicação de um *algoritmo de identificação*, o qual a partir da amostras das dinâmicas de um sistema, fornece uma boa aproximação das componentes da função de transição na forma das bases dos operadores correspondentes. A identificação foi feita com e sem a aplicação da *restrição de envelope dinâmico*, que impõe limites inferiores e superiores para as dinâmicas do sistema. Posteriormente, foi avaliada a qualidade da identificação medindo a distância entre as dinâmicas do sistema original (ideal) e o identificado. A comparação dos erros cometidos na identificação com e sem a aplicação da *restrição de envelope dinâmico* mostrou que este tipo de restrições pode ser benéfica para a melhora da precisão da identificação nos casos em que se dispõe de um pequeno número de amostras.

Com o objetivo de desenvolver técnicas de modelagem da informação conhecida a priori acerca de um sistema, foi efetuada a *modelagem* e *simulação* de um ciclo celular hipotético como um sistema dinâmico finito de reticulados. Ele conseguiu representar adequadamente as suposições prévias acerca do seu comportamento dinâmico. Vemos assim, que os FLDS constituem um modelo hipotético que é capaz de incorporar características do sistema real.

Pode apreciar-se a utilidade deste tipo de técnicas de modelagem, simulação e identificação usadas interativamente, da seguinte maneira. Um modelo pode ser criado baseado em conhecimento prévio do sistema real. Simulando este sistema podem inferir-se novas

características sobre a lógica e os mecanismos que regem a sua dinâmica, assim como, pode obter-se informação que permita aperfeiçoar o modelo e simulá-lo novamente. Como conseqüência desta interação sucessiva com o simulador, vai-se aumentando a compreensão e a informação que se possui sobre o funcionamento do sistema dinâmico. Quando o modelo resulte satisfatório, ele pode ser flexibilizado criando um envelope dinâmico, que represente um limite inferior e superior para as dinâmicas do sistema, para então efetuar a identificação, a partir de amostras do sistema real, restrita a este envelope dinâmico. Assim, esta técnica permite a integração de conhecimento prévio à informação presente nos dados reais, o que é muito importante quando o número de amostras disponíveis é escasso comparado com o necessário para obter uma boa precisão na identificação, situação que é muito freqüente na prática.

1.3 Estrutura da Dissertação

1.3.1 Conteúdo

- No Capítulo 2, **Representação em Reticulados**, se apresentam os conceitos básicos de álgebra de reticulados, álgebra booleana e representação canônica de operadores. Apresentamos lá a estrutura algébrica sobre a qual é feita a modelagem, simulação e identificação de sistemas dinâmicos finitos.

- No Capítulo 3, **Sistemas Dinâmicos Finitos**, se apresentam os sistemas dinâmicos finitos, sistemas livres e sistemas dinâmicos finitos de reticulados (FLDS: "Finite Lattice Dynamical Systems"), sistemas Booleanos e se dá a decomposição da função de transição num vetor de operadores de reticulados representado pelo conjunto de suas

bases. Se dá um exemplo de um sistema dinâmico Booleano simples, e de sua simulação.

Se apresentam a *modelagem* e *simulações* realizadas correspondentes a um *ciclo celular* hipotético que representa características específicas de sistemas biológicos. Elas foram efetuadas em valores binários e em 5 níveis. Apresenta-se uma descrição do comportamento obtido do ciclo celular e mostra-se um resumo das simulações efetuadas em diversos experimentos.

- No Capítulo 4, **Identificação de Sistemas Livres**, apresenta-se os fundamentos da teoria de aprendizado computacional, base teórica da identificação de sistemas efetuada. É descrito o algoritmo ISI de particionamento sucessivo de intervalos, utilizado no aprendizado. Descreve-se a metodologia da identificação definindo-se os conceitos de *algoritmo de aprendizado*, *amostra de treinamento*, *função de perda*, *erro* e *aprendizado PAC*. São apresentados exemplos da identificação de sistemas e da avaliação dos erros cometidos.

- No Capítulo 5, **Identificação com Restrição de Envelope**, apresenta-se o conceito de *restrição de envelope dinâmico* e a sua aplicação na melhora da precisão na identificação de sistemas dinâmicos finitos. Mostra-se resultados experimentais que exemplificam o seu potencial.

- No Capítulo 6, **Conclusão**, discute-se os resultados obtidos referentes à modelagem e simulação do ciclo celular e identificação de sistemas dinâmicos com aplicação de restrição de envelope dinâmico. Discute-se a utilidade destas técnicas, possíveis aplicações e futuras linhas de pesquisa.

- O Apêndice A, **Modelo Binário do Ciclo Celular**, apresenta a arquitetura do ciclo celular binário e a definição completa das funções de transição e realimentação.

- O Apêndice B, **Modelo em Níveis de Cinza do Ciclo Celular**, apresenta a arquitetura do ciclo celular em cinco níveis de intensidade e a definição completa das funções de transição e realimentação.

- No Apêndice C, **Árvores de Classificação por Multirresolução** apresenta-se a técnica desenvolvida para a geração automática de árvores de classificação por multirresolução, os resultados experimentais obtidos, e discute-se sua utilidade como técnica de aprendizado, a qual é também factível de ser utilizada para a identificacão de sistemas dinâmicos finitos.

1.3.2 Contribuição

As contribuições do presente trabalho encontram-se nos Capítulos 3, 4 e 5 e nos Apêndices A, B e C, e consistem na modelagem e simulação de ciclo celular, identificação de sistemas dinâmicos finitos de reticulados, aplicação de restrição de envelope dinâmico, e a técnica de geração automática de árvores de classificação por multirresolução.

Capítulo 2

Representação em Reticulados

2.1 Álgebra de Reticulados

Dado um conjunto L, uma relação binária $\leq$ em L é chamada de uma *relação de ordem parcial* se ela é

- *Reflexiva:* $\forall x \in L, \quad x \leq x$
- *Anti-simétrica:* $\forall x, y \in L, \quad \text{se} \quad x \leq y \quad \text{e} \quad y \leq x, \quad \text{então} \quad x = y$
- *Transitiva:* $\forall x, y, z \in L, \quad \text{se} \quad x \leq y \quad \text{e} \quad y \leq z, \quad \text{então} \quad x \leq z$

Dizemos que $(L, \leq)$, ou simplesmente L, é um *conjunto parcialmente ordenado* ou *poset* ("partially ordered set") [18].

Dados $a, b \in L$ e $X \subseteq L$, dizemos que b é um *limitante superior* de X, se para todo $x \in X$ temos que $x \leq b$ e que a é um *limitante inferior* de X, se para todo $x \in X$, temos que $a \leq x$.

O *supremo* de X em L, se existir, é o menor limitante superior de X, em outras palavras, um limitante superior x de X, tal que $x \leq u$, para qualquer outro limitante

superior u de X.

O *ínfimo* de X em L, se existir, é o maior limitante inferior de X.

Pela anti-simetria de $\leq$, o supremo e o ínfimo de X em $(L, \leq)$ são únicos, sempre que existem.

O supremo e o ínfimo de dois elementos x e y de L, se existem, são denotados respectivamente por $x \vee y$ e $x \wedge y$. O supremo de X em L é denotado por $\vee X$ enquanto o ínfimo de X em L por $\wedge X$.

Se $\vee X \in X$, então $\vee X$ é dito ser o *elemento máximo* de X. Reciprocamente, se $\wedge X \in X$, então $\wedge X$ é o *elemento mínimo* de X.

Um elemento de um poset é dito *minimal* (*maximal*) se nenhum outro elemento do poset é menor (maior) do que ele.

Um subconjunto X de um poset L é dito um *intervalo fechado* de L se e somente se existem dois elementos a e b em L tais que, para todo $x \in L$,

$$a \leq x \leq b \Leftrightarrow x \in X.$$

Tal intervalo fechado é denotado por $[a, b]$ e chamamos a e b, respectivamente, os seus *extremos esquerdo* e *direito*.

O conjunto de intervalos em L, com a relação de ordem parcial $\leq$, definida por

$$[a, b] \leq [a', b'] \quad \Leftrightarrow \quad a' \leq a \quad \text{e} \quad b \leq b'$$

constitui um poset.

Dizemos que um poset L é um *reticulado completo* se todo subconjunto não vazio X de L tem um ínfimo e um supremo em L.

Os *limitantes universais* de um reticulado completo L são o *maior elemento* I e o *menor elemento* O, satisfazindo $O \leq x \leq I$ para todo $x \in L$.

A existência e unicidade deles segue das igualdades $I = \vee L$ e $O = \wedge L$.

Um reticulado L é dito *distributivo* se para quaisquer três elementos $a, b, c \in L$ as propriedades distribuitivas

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \tag{2.1}$$

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \tag{2.2}$$

valem; caso contrario é dito *não distribuitivo*.

Sejam $a, b \in L$ tais que $a \leq b$. O mapeamento $\lambda_{a,b}$ de L em $\{0, 1\}$ dado por

$$\lambda_{a,b}(x) = 1 \quad \Leftrightarrow \quad x \in [a, b], \quad \forall x \in L,$$

é chamado *operador binário sup-gerador*.

Seja $P(L)$ o conjunto das partes de L, seja $Fu[L, P(L)]$ o conjunto de funções de L em $P(L)$ e seja ψ um mapeamento (também chamado *operador de reticulado*) definido sobre L. O *núcleo* ou *kernel* $K(\psi)$ de ψ é a função em $Fun[L, P(L)]$ dada por [17]

$$K(\psi)(y) = \{x \in L : y \leq \psi(x)\}, \quad \forall y \in L.$$

Seja $Max(L)$ o conjunto de elementos maximais do poset L. Dizemos que o poset L tem um *envelope maximal* se, para todo $x \in L$, existe $y \in Max(L)$ tal que $x \leq y$. Todo poset finito L tem um envelope maximal $Max(L)$.

A *base* $B(\psi)$ do *operador de reticulados* ψ é a função dada por

$$\mathbf{B}(\psi)(y) = Max(\{[a,b] : [a,b] \subseteq K(\psi)(y)\}), \quad \forall y \in L.$$

Seja L um reticulado finito e ψ um operador definido sobre L, então

$$\psi(x) = \vee\{y \in L : \vee\{\lambda_{a,b}(x) : [a,b] \in \mathbf{B}(\psi)(y)\}\}$$

para todo $x \in L$, e ψ é dito de ter uma decomposição minimal por um conjunto de operadores binários sup-geradores.

2.2 Funções Booleanas

2.2.1 Álgebra Booleana

Uma álgebra Booleana consiste de um conjunto de elementos B, junto com duas operações binárias $\{\bullet\}$ e $\{+\}$ e uma operação unária $\{\overline{\bullet}\}$, tais que são válidos os seguintes axiomas [9, 10, 12]:

1. O conjunto B contém pelo menos dois elementos a, b tais que $a \neq b$.

2. *Fechamento*: Para cada a, b em B,

a. $a + b$ está em B

b. $a \bullet b$ está em B

3. *Leis comutativas*: Para cada a, b em B,

a. $a + b = b + a$

b. $a \bullet b = b \bullet a$

4. *Leis associativas*: Para cada a, b, c em B,

a. $(a + b) + c = a + (b + c) = a + b + c$

b. $(a \bullet b) \bullet c = a \bullet (b \bullet c) = a \bullet b \bullet c$

5. *Identidades*:

a. Existe um elemento identidade com respeito a $+$, designado por 0, tal que $a + 0 = a$ para cada a em B.

b. Existe um elemento identidade com respeito a $\bullet$, designado por 1, tal que $a \bullet 1 = a$ para cada a em B.

6. *Leis distributivas*: Para cada a, b, c em B,

a. $a + (b \bullet c) = (a + b) \bullet (a + c)$

b. $a \bullet (b + c) = (a \bullet b) + (a \bullet c)$

7. *Complemento*: Para cada a em B, existe um elemento $\overline{a}$ em B (o complemento de a) tal que

a. $a + \overline{a} = 1$

b. $a \bullet \overline{a} = 0$

Outras propriedades algébricas importantes estão dadas pelos seguintes teoremas:

8. Teorema de *Idempotência*:

   a. $a + a = a$

   b. $a \bullet a = a$

9. Teorema de *Involução*:

   $\overline{\overline{a}} = a$

10. Teoremas de *Simplificação*:

    a. $a \bullet b + a \bullet \overline{b} = a, \qquad (a + b) \bullet (a + \overline{b}) = a$

    b. $a + a \bullet b = a, \qquad a \bullet (a + b) = a$

    c. $(a + \overline{b}) \bullet b = a \bullet b, \qquad (a \bullet \overline{b}) + b = a + b$

11. Leis de *DeMorgan*:

    a. $\overline{(a + b + c + \ldots)} = \overline{a} \bullet \overline{b} \bullet \overline{c} \bullet \ldots$

    b. $\overline{(a \bullet b \bullet c \bullet \ldots)} = \overline{a} + \overline{b} + \overline{c} + \ldots$

Exemplo de álgebra Booleana: O conjunto $B = \{0, 1\}$ com as operações $\bullet$, $+$ e $\overline{\bullet}$, definidas na seguinte tabela, é uma álgebra Booleana:

a	b	$a+b$	$a \bullet b$	$\overline{a}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Tabela 2.1: Definição das operações $\bullet$, $+$ e $\overline{\bullet}$ em $B = \{0, 1\}$

2.2.2 Álgebras Booleanas como Reticulados

Uma álgebra booleana é um caso particular de reticulado com as seguintes propriedades.

Em um reticulado $[L, \vee, \wedge]$, o *complemento* de um elemento $a \in L$ é um elemento b tal que

$$a \wedge b = O \quad \text{e} \quad a \vee b = I.$$

Um reticulado $[L, \vee, \wedge]$ é dito *complementado* se para cada elemento $a \in L$ existe um complemento $b \in L$. Em um reticulado $[L, \vee, \wedge]$ distributivo, os complementos são únicos quando eles existem.

Uma *álgebra booleana* ou *reticulado booleano* é um reticulado distributivo e complementado.

2.2.3 Expressões Booleanas

Sejam $a_1, a_2, ..., a_n$ variáveis que tomam valor em B. Uma expressão Booleana é qualquer expressão construída a partir destas variáveis aplicando-se um número finito de vezes as operações $+$, $\bullet$ e $\overline{\bullet}$ e as constantes em B [12].

Exemplos de expressões Booleanas:

Nas seguintes expressões o operador produto foi omitido.

$$\overline{a_1 a_2} + a_1(\overline{a_2} + a_2\overline{a_3}), \quad \overline{a_1 a_2} + a_1\overline{a_2} + a_1 a_2\overline{a_3}, \quad a_2 + a_1\overline{a_3}$$

■

Duas expressões Booleanas são equivalentes se e somente se uma delas pode ser deduzida da outra através da aplicação das leis ou teoremas da álgebra Booleana. A simplificação de expressões é a obtenção de uma expressão equivalente com menor número de produtos envolvendo um número menor de literais (variáveis ou o seu complemento).

2.2.4 Forma Canônica de Expressões Booleanas

Um produto em que aparecem n variáveis, no qual a aparece se, e somente se, o seu complemento $\overline{a}$ não aparece, é denominado *produto canônico* (ou *mintermo*). Analogamente, uma soma na qual a aparece se, e somente se, o seu complemento $\overline{a}$ não aparece é denominada *soma canônica* (ou *maxtermo*) [19].

Qualquer função canônica pode ser expressa em termos de soma de produtos canônicos (*forma normal disjuntiva* ou *FND*) ou em termos de produto de somas canônicas (*forma normal conjuntiva* ou *FNC*).

Exemplos:

- $\overline{a}_1\overline{a}_2\overline{a}_3 + \overline{a}_1\overline{a}_2 a_3 + a_1\overline{a}_2\overline{a}_3 + a_1\overline{a}_2 a_3 + a_1 a_2\overline{a}_3$: *soma de produtos canônicos (FND)*

- $(a_1 + \overline{a_2} + a_3)(a_1 + \overline{a_2} + \overline{a_3})(\overline{a_1} + \overline{a_2} + \overline{a_3})$: *produto de somas canônicas (FNC)*

2.2.5 Funções Booleanas:

Uma função Booleana é uma função de n variáveis de $\{0,1\}^n$ em $\{0,1\}$. Essas n variáveis, denotadas por $x_1, x_2, \ldots, x_n$, são denominadas variáveis Booleanas [19].

Representação Cúbica de uma Função Booleana:

As seqüências de n "bits", correspondentes a todas as possíveis combinações de valores que as n variáveis de uma função Booleana podem tomar, podem ser representadas por pontos no *n-espaço*. A coleção de todos os 2^n pontos possíveis formam os vértices de um *n-cubo*.

Dizemos que um k-cubo tem *dimensão k*. Dado um conjunto V de vértices de um n-cubo, dizemos que um k-kubo formado pelos vértices de V é *maximal* se não existe outro cubo de dimensão maior também formado por vértices de V e que contenha o k-cubo.

A *representação cúbica* de uma função Booleana de n variáveis consiste do conjunto de vértices do *n-cubo* que correspondem aos mintermos de f.

A *representação cúbica minimal* de uma função Booleana de n variáveis consiste do conjunto de todos os cubos maximais formados por vértices do n-cubo que correspondem aos seus mintermos.

O n-cubo forma uma estrutura de reticulado Booleano. Portanto, podemos associar a um cubo a noção de *intervalo*, cujo extremo esquerdo e direito são respectivamente o menor e o maior elemento do cubo. A *dimensão* de um intervalo [A,B], denotado $dim([A,B])$, é

definida como a dimensão do cubo associado a ele, que é também equivalente a $\|B\| - \|A\|$, onde $\|A\|$ representa o número de bits iguais a 1 em A.

Uma expressão Booleana é considerada uma *expressão minimal* se:

1. não existe nenhuma outra expressão equivalente com um número menor de termos

e

2. não existe nenhuma outra expressão equivalente com igual número de termos mas com menor número de variáveis.

Capítulo 3

Sistemas Dinâmicos Finitos

3.1 Sistemas Dinâmicos Finitos de Reticulados

Sejam X, U e Y conjuntos finitos, N um inteiro positivo, ϕ e ψ mapeamentos de $X^{N+1} \times U^{N+1}$ em X. Um *Sistema Dinâmico Finito* ou *FDS* (Finite Dinamical System) $S(\phi)$ é dado por, para todo $t \geq N$,

$$x[t+1] = \phi(x[t], x[t-1], \ldots, x[t-N], u[t], \ldots, u[t-N])$$

$$y[t] = \psi(x[t], \ldots, x[t-N], u[t], \ldots, u[t-N])$$

onde $x[t] \in X$, $u[t] \in U$, e $y[t] \in Y$ para todo $t \geq 0$. Os mapeamentos ϕ e ψ são chamados respectivamente, *função de transição* e *função de saída*. As variáveis t, x, u e y são chamadas *tempo discreto*, *estado*, *entrada* e *saída*, respectivamente. O conjunto X é chamado *espaço de estados* e N *memória do sistema* [2].

Na prática podemos ter funções de transição e saída que dependam de um número diferente de estados e entradas passadas; no entanto, para manter a notação simples, as

escrevemos dependendo dos $N+1$ estados e entradas mais recentes.

Os sistemas definidos como acima são invariantes por translação de tempo, isto é, a função de transição é a mesma para todo tempo discreto t.

A Figura 3.1 mostra uma representação gráfica de um sistema dinâmico finito (FDS) de memória N:

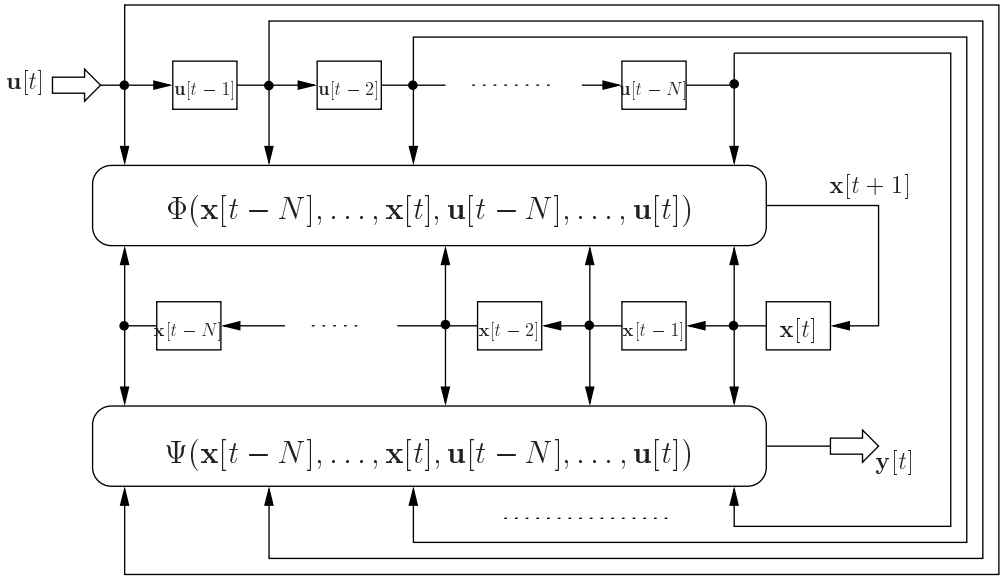


Figura 3.1: Sistema Dinâmico Finito de Memória N (extraído de [2]).

Um sistema dinâmico finito pode ser representado equivalentemente por

$$x[t+1] = \varphi(x[t], u[t])$$

$$y[t] = \xi(x[t], u[t])$$

onde $x[t] \in X^{N+1}$ e $u[t] \in U^{N+1}$.

Se não existe entrada (ou, equivalentemente, se temos uma entrada constante) –e ξ é

a função identidade– chamamos o sistema ϕ dado por, para todo $t \geq 0$,

$$x[t+1] = \phi(x[t])$$

$$y[t] = x[t]$$

um *sistema livre*. A seqüência $x[0], x[1], \ldots, x[t], \ldots$ é chamada uma *dinâmica* (ou órbita) do sistema ϕ.

Se o espaço de estados X é um reticulado finito, então o sistema dinâmico finito livre pode ser representado pela base da função de transição e é chamado *Sistema Dinâmico Finito de Reticulados* (FLDS: Finite Lattice Dynamical System) *livre de entradas*. Em particular, se X é dado pelo produto Cartesiano de cadeias finitas (i.e., conjuntos completamente ordenados finitos) K, ou seja, $X = K^n$, então a função de transição ϕ pode ser decomposta num vetor de n funções de transição, com componentes ϕ_j de X em K. Dado que ϕ_j é um operador de reticulados, ele tem uma base e uma correspondente representação canônica. Assim, a função de transição ϕ pode ser representada pelo conjunto de bases $\{\mathbf{B}(\phi_j) : j \in \{1, 2, \ldots, n\}\}$.

Dizemos que um FLDS é *Booleano* se $K = \{0, 1\}$ e X é o conjunto de vetores binários de comprimento n. Num FLDS Booleano, cada função de transição é uma função Booleana.

A *arquitetura* de um sistema dinâmico finito indica as dependências mais relevantes entre as variáveis (genes) em tempo e espaço, ou seja, o valor de quais variáveis (nível de expressão dos genes) e em quais instantes de tempo influem mais fortemente no valor de cada variável do sistema.

Nas simulações de sistemas dinâmicos finitos de reticulados realizadas no presente trabalho foi utilizado o *Simulador para Redes de Expressão Gênica SGEN* ("Simulator for Gene Expression Networks") desenvolvido pelo Professor do IME - USP Dr. Marco Dimas Gubitoso [20] no contexto do projeto CAGE ("Cooperation for Analysis of Gene Expression").

3.2 Um Sistema Booleano Simples

Consideramos um FLDS Booleano $S(\Phi)$ em $X = \{0,1\}^5$. Um estado deste sistema é um elemento $x \in X$. Cada componente x_j, $j \in \{1,2,3,4,5\}$, do estado depende dos seus estados passados e atual como assim também dos estados passados e atual de dois outros componentes. Cada componente tem um comportamento cíclico, isto é, ele toma o valor 1 somente se o seu valor num certo número de instantes imediatamente anteriores foi 0. Além disso, outros dois elementos podem interagir sobre ele, de modo a aumentar o seu ciclo. A rede de interação tem uma arquitetura em forma de estrela (Figura 3.2).

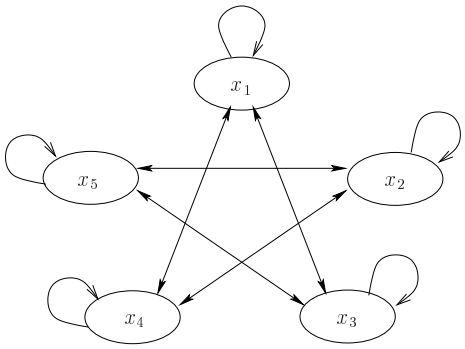


Figura 3.2: Arquitetura do Sistema.

As seguintes regras definem a função de transição dos componentes do sistema:

- O componente x_1 toma o valor 1 se: (a) o seu valor no estado atual é 0 (isto significa

que antes de tomar o valor 1 ele necessariamente toma o valor 0) e (b) existe pelo menos um 1 nos três últimos instantes dos componentes x_3 e x_4, ou o valor dos componentes x_3 e x_4 foi 0 nos últimos cinco instantes.

- O componente x_2 toma o valor 1 se: (a) o seu valor nos dois últimos instantes foi 0 (isto significa que antes de tomar o valor 1 ele necessariamente toma o valor 0 pelo menos em dois instantes prévios) e (b) existe pelo menos um 1 nos componentes x_4 ou x_5 nos últimos três instantes, ou ambos os componentes foram 0 nos últimos 6 instantes.

- O componente x_3 toma o valor 1 se: (a) o seu valor nos últimos três instantes foi 0 (isto significa que antes de tomar o valor 1 ele necessariamente toma o valor 0 pelo menos nos três instantes prévios) e (b) existem pelo menos dois 1 nos últimos 5 instantes do componente x_1, ou pelo menos um 1 nos últimos 3 instantes do componente x_5, ou o componente x_5 foi zero nos últimos cinco instantes.

- O componente x_4 toma o valor 1 se (a) o seu valor nos últimos quatro instantes foi 0 (isto significa que antes de tomar o valor 1 ele necessariamente toma o valor 0 pelo menos em quatro instantes precedentes) e (b) igual que o componente x_1, mas com relação aos componentes x_1 e x_2, ou os último quatro instantes de ambos os componentes foram 0.

- O componente x_5 toma o valor 1 se (a) o seu valor nos últimos cinco instantes foi 0 (isto significa que antes de tomar o valor 1 ele necessariamente toma o valor 0 em pelo menos os cinco instantes precedentes) e (b) existe pelo menos um 1 nos últimos quatro instantes dos componentes x_2 e x_3, ou um deles é 0 nos últimos 6 instantes.

Estas regras podem ser formuladas mediantes funções Booleanas.

A Figura 3.3 mostra a simulação do sistema para as duas condições iniciais. Depois

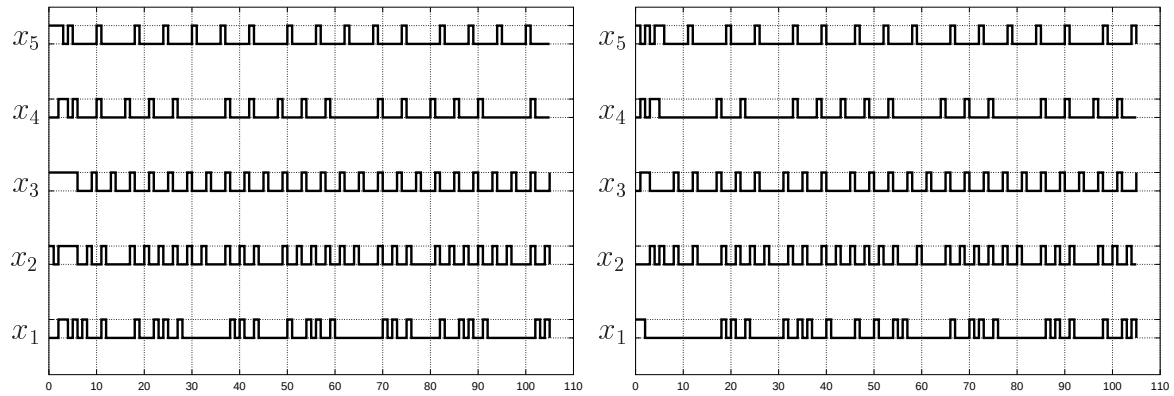


Figura 3.3: Dois Resultados da Simulação.

de alguns instantes, o sistema entra num ciclo limite, ou seja, um conjunto de vetores de estado que são repetidos ciclicamente. Na primeira simulação (esquerda) o sistema entra no ciclo limite no instante 9, enquanto na segunda (direita) ele entra no instante 60.

3.3 Simulação de Ciclo Celular

Dada a complexidade das redes biológicas, a modelagem, simulação e visualização são ferramentas importantes para o entendimento do comportamento dinâmico dos sistemas e para comparar os resultados da modelagem matemática com os dados experimentais. Da simulação de modelos ideais podem discernir-se características dinâmicas que levem a uma maior compreensão teórica das redes genéticas.

A modelagem baseada em conhecimento biológico prévio pode ser utilizada para estabelecer restrições que melhorem a qualidade da identificação de sistemas dinâmicos a partir de dados reais.

No desenvolvimento deste tipo de técnicas, foi modelada e simulada a dinâmica de controle da rede de expressão gênica correspondente a um ciclo celular hipotético. A modelagem das interações entre todos os genes da rede foi feita de maneira de obter um sistema que se comporte de forma similar às hipóteses sobre o comportamento biológico no referente a robustez, flexibilidade, adaptabilidade, etc..

Para realizar as simulações da dinâmica do sistema foi necessário estudar os princípios de funcionamento e forma de uso do Simulador para Redes de Expressão Gênica *SGEN: "Simulator for Gene Expression Networks"* [20]. As dificuldades encontradas no uso do simulador foram consideradas na melhora do projeto deste.

O ciclo celular foi modelado como um sistema dinâmico finito de reticulados. Numa primeira etapa, o sistema foi modelado com valores binários, e posteriormente ele foi modelado com funções multivaloradas, podendo seus elementos assumir cinco valores ou níveis de cinza: -2, -1, 0, 1, 2 (Seções A.2 e B.2).

Isto foi feito para explorar as diferenças de complexidade entre o sistema multivalorado e o sistema binário e averiguar quanto podemos aprender sobre um sistema multivalorado a partir do estudo (modelagem) de um sistema binário. Conseguimos produzir padrões de sinal muito semelhantes nos modelos binário e multivalorado ao procurar modelar determinadas características da dinâmina do sistema. Do ponto de vista de identificação de sistemas, é conveniente escolher o sistema mais simples que seja suficiente para descrever o fenômeno de interesse. Essa complexidade, diretamente relacionada ao número de trajetórias possíveis do sistema, afeta diretamente os erros de estimação. Quanto menor a complexidade dos sistemas considerados, menores tendem a ser os erros de estimação de parâmetros dos sistemas (a partir de trajetórias observadas).

A modelagem para sistemas multivalorados (níveis de cinza) engloba uma dificuldade adicional decorrente do maior número de graus de liberdade. Neste processo, foi sugerida uma mudança na especificação da entrada do simulador, dado que a representação anteriormente utilizada (árvore de intervalos) tinha uma complexidade da ordem de 5^n, onde n é o número de argumentos da função considerada.

3.3.1 Descrição do comportamento do ciclo celular modelado

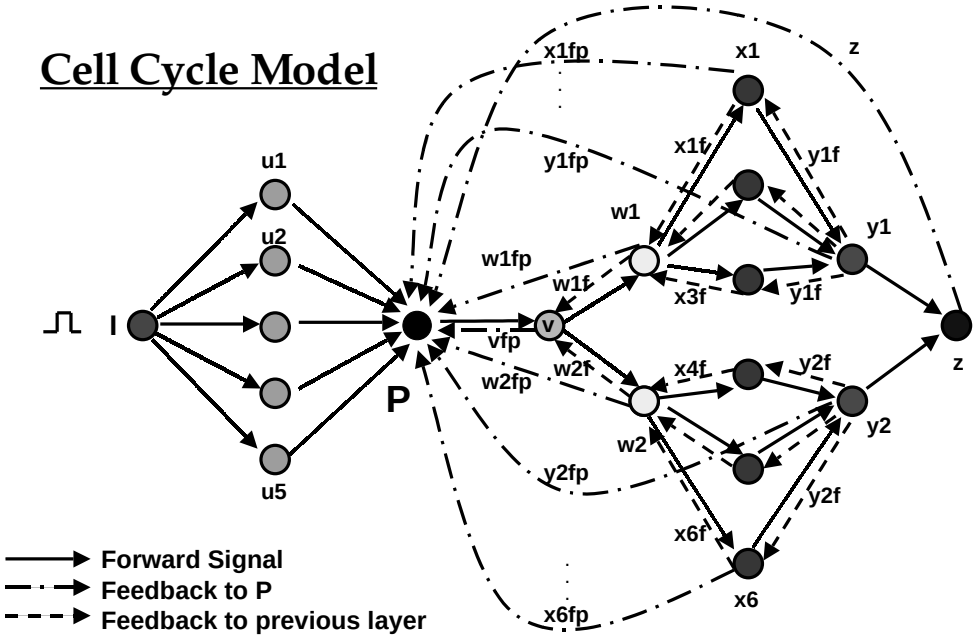


Figura 3.4: Diagrama da Arquitetura do Ciclo Celular.

O ciclo consta de seis etapas na sua parte principal, representadas pela expressão das camadas de genes: $\boldsymbol{P}$, $\boldsymbol{v}$, $\boldsymbol{w}$, $\boldsymbol{x}$, $\boldsymbol{y}$ e $\boldsymbol{z}$, e de uma etapa prévia ou de excitação, constituída pela camada de genes $\boldsymbol{u}$ que recebem o estímulo externo $\boldsymbol{I}$ (Figura 3.4).

No funcionamento normal do sistema, um sinal externo I excita os genes u, e eles se expressam quando o estímulo resulta suficiente e existe a quantidade de substância necessária, o que é indicado no modelo por determinados valores gerados aleatoriamente que debem superar determinados limiares. Os sinais provenientes dos genes u são integrados

em P e, em condições favoráveis, este gene se expressa iniciando um ciclo celular. O sinal vai se propagando pelas camadas seguintes de genes e o sistema tende a impedir que um novo ciclo celular seja realizado enquanto não termine o atual.

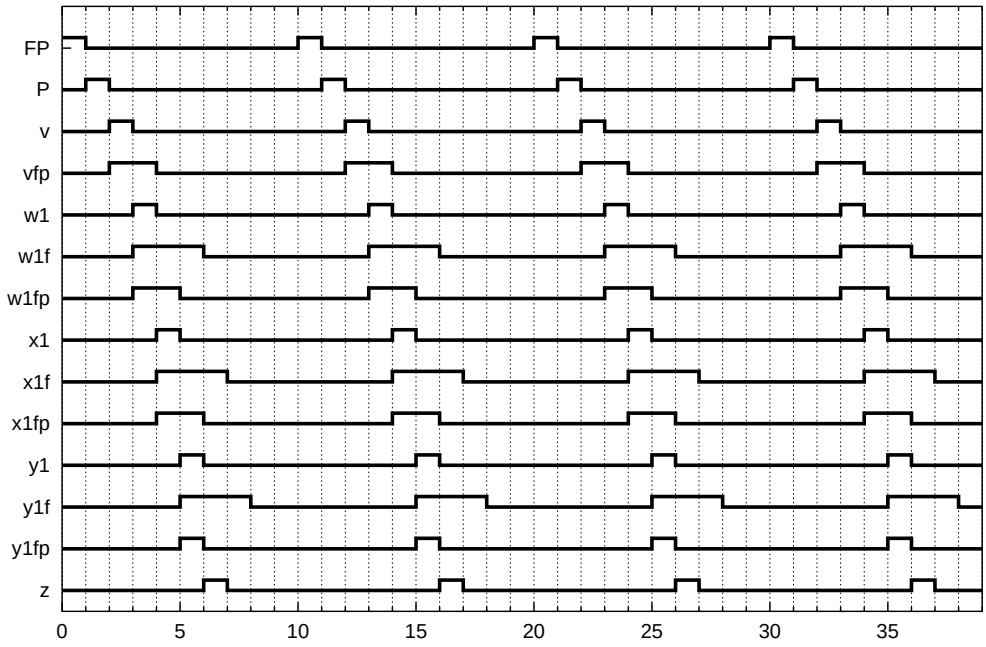
A expressão de cada um dos genes u depende da intensidade do estímulo externo I e da existência de uma quantidade de substância necessária, o que é indicado por um valor positivo N associado a esse gene. Este valor é dado para cada gene u_i, $1 \leq i \leq 5$, por três valores aleatórios: n_{i1}, n_{i2} e n_{i3} que devem superar três limiares: t_{i1}, t_{i2} e t_{i3} (Seções A.2.1 e B.2.1). Estas e as demais funções de transição e realimentação estão especificadas nos apêndices.

Os sinais provenientes da camada de genes u são integrados em P dando como resultado o sinal FP. O ciclo celular se inicia com a expressão do gene P. O valor desta expressão depende da intensidade de FP e das *realimentações* que P recebe das camadas posteriores de genes (Seções A.2.2 e B.2.2). Se existe sinal se propagando pelas camadas posteriores ao gene P, será necessária uma intensidade maior de FP para atingir um nível de expressão em P. A intensidade requerida para a expressão de P quando um sinal já está se propagando pelas camadas posteriores depende do lugar no ciclo em que aquele sinal está. A maior intensidade requerida ocorre quando o sinal estiver atravessando a camada x. Será algo menor quando se encontra em w ou y, menor ainda quando estiver em v ou z. O menor valor de intensidade requerida para a expressão de P será quando não exista sinal algum se propagando pelas camadas posteriores.

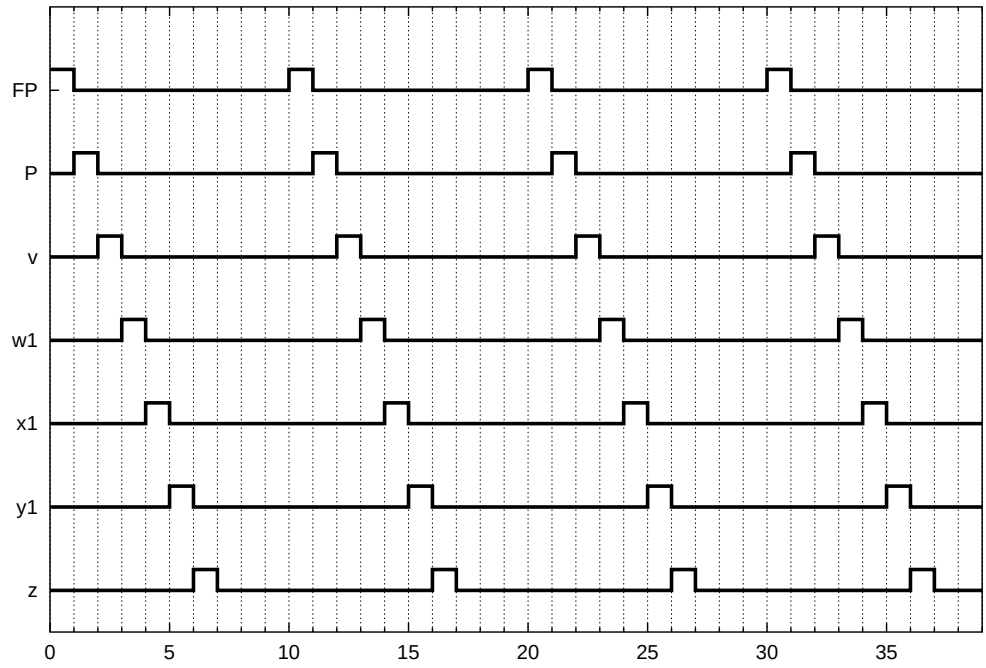
Cada gene das camadas seguintes se expressa dependendo da excitação que recebe e das realimentações que chegam a ele da camada seguinte. Quando um destes genes se expressa, envia também um sinal de realimentação para P e outro para a camada anterior

de genes que torna mais difícil a expressão dos genes prévios por algum tempo. Mesmo assim, se eles forem excitados com um sinal muito forte podem se expressar.

As Figuras 3.5, 3.6, 3.7 e 3.8 (Modelo Binário) e as Figuras 3.9 e 3.10 (Modelo em Níveis de Cinza), mostram o comportamento do ciclo celular excitado por um sinal FP periódico de freqüência crescente. Quando o período de FP é maior ou igual ao período do ciclo celular, o sistema tem o seu funcionamento normal (Figuras 3.5, 3.6, 3.9 e 3.10). Quando o período (decrescente) de FP é menor que o período do ciclo celular, o sistema tenta rejeitar o sinal excessivo e manter o seu período natural, até que um sinal muito forte em freqüência ou amplitude (caso níveis de cinza) consegue vencer a resistência do sistema dada pelas realimentações entre camadas e as realimentações para o primeiro gene do ciclo.



(a) Sinais representativos a través do Ciclo Celular.



(b) Comportamento do sistema.

Figura 3.5: FP = Oscilador de período 10 (Modelo Binário).

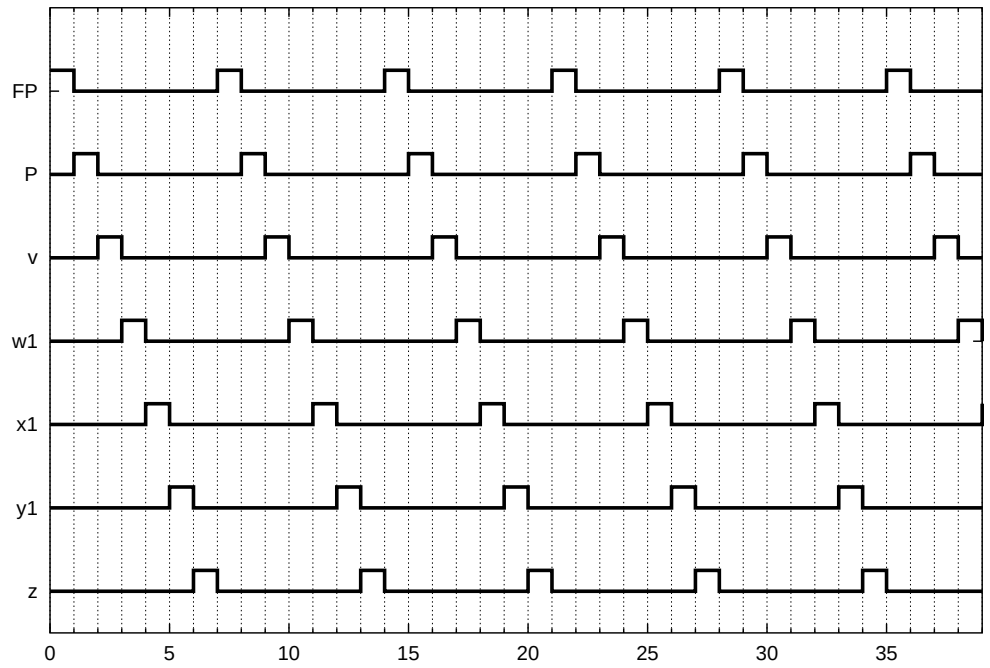


Figura 3.6: Comportamento do sistema com FP = Oscilador de período 7 (Modelo Binário).

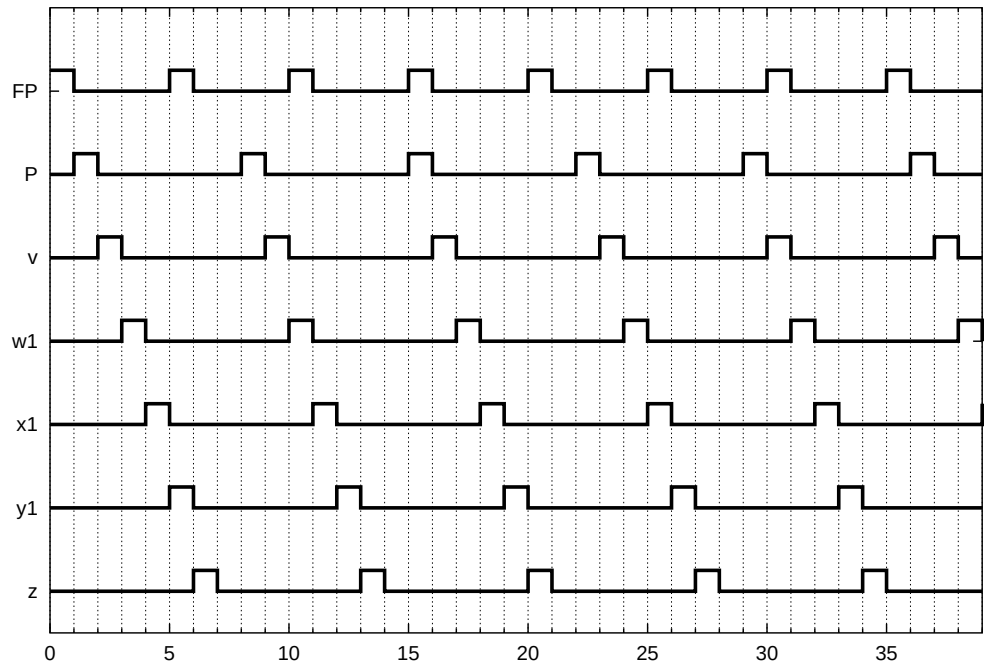


Figura 3.7: Comportamento do sistema com FP = Oscilador de período 5 (Modelo Binário).

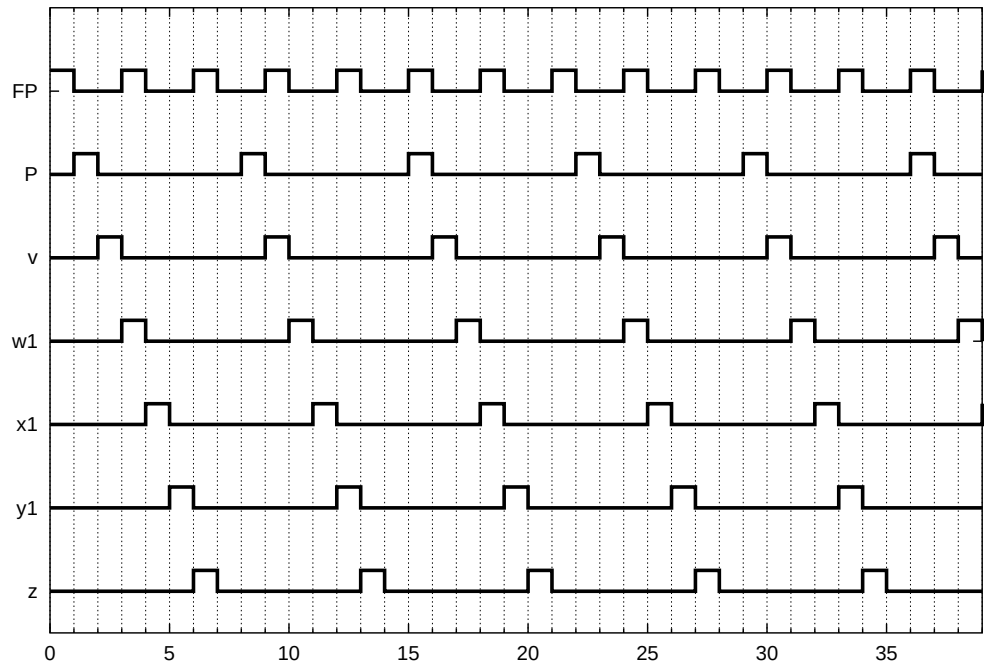
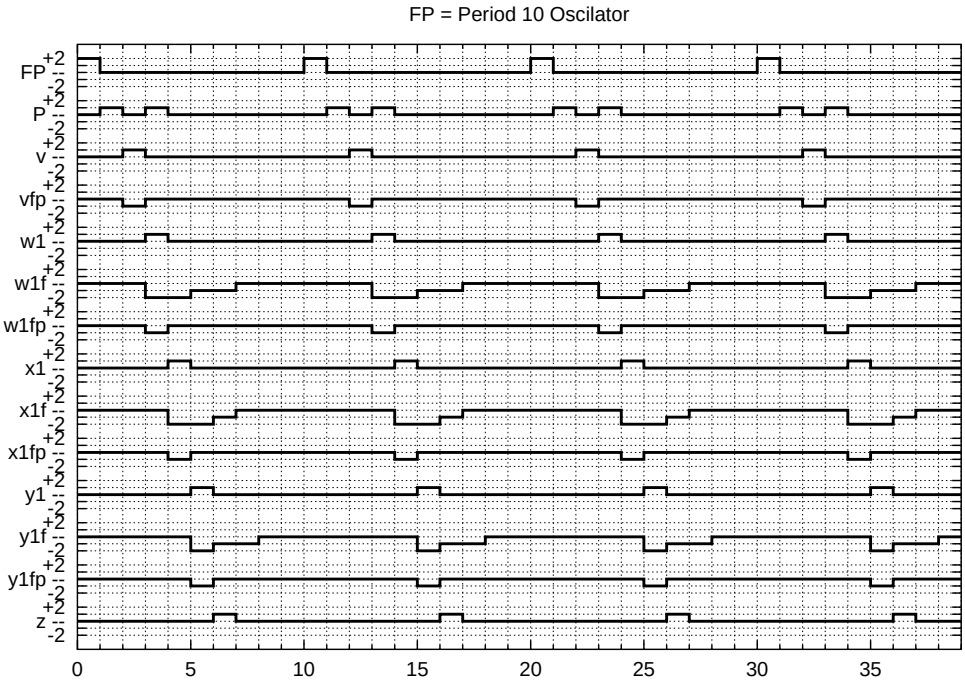
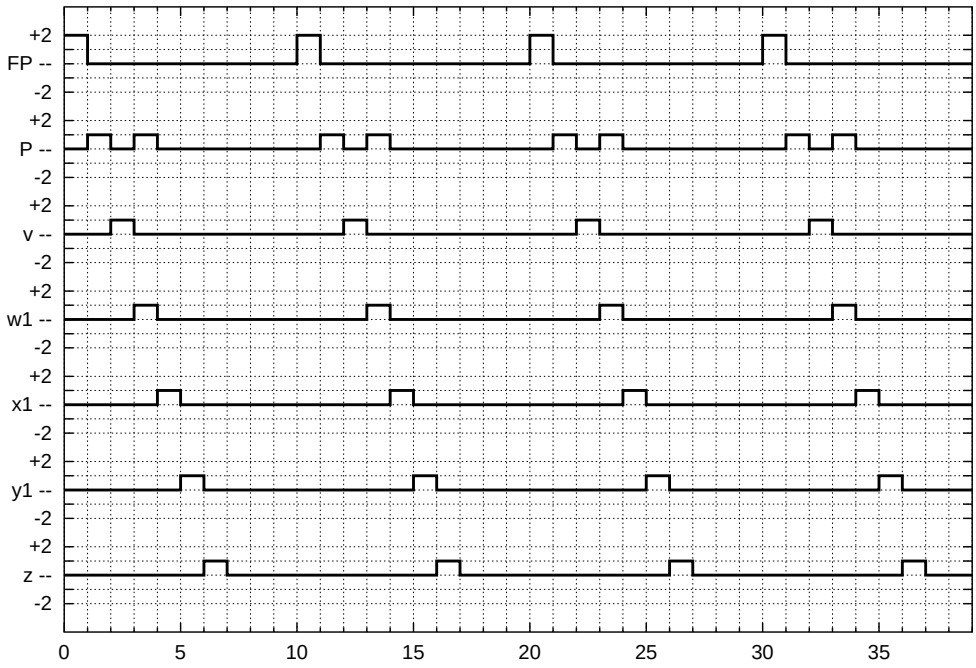


Figura 3.8: Comportamento do sistema com FP = Oscilador de período 3 (Modelo Binário).



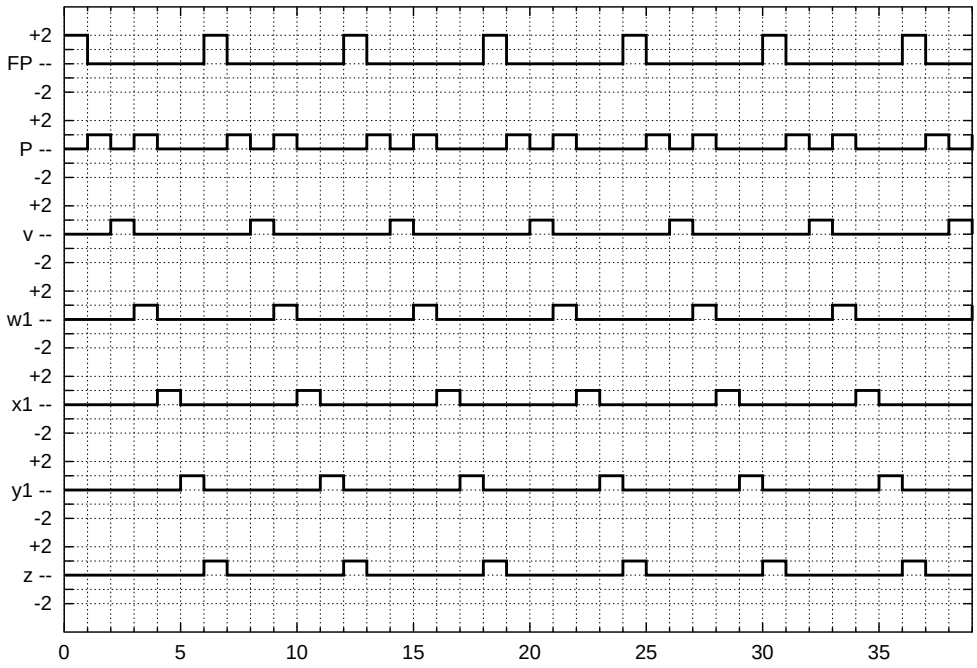
(a) Sinais representativos a través do Ciclo Celular.



(b) Comportamento do sistema.

Figura 3.9: FP = Oscilador de período 10 (Modelo em Níveis de Cinza).

(a) Sinais representativos a través do Ciclo Celular.



(b) Comportamento do sistema.

Figura 3.10: FP = Oscilador de período 6 (Modelo em Níveis de Cinza).

Se um sinal FP suficientemente forte (em freqüência ou amplitude) consegue a expressão de P quando existe sinal se propagando nas camadas posteriores, as realimentações para a camada anterior das seguintes etapas estarão ativas e resistirão à passagem deste novo sinal (Figuras 3.11, 3.12, 3.13 e 3.14).

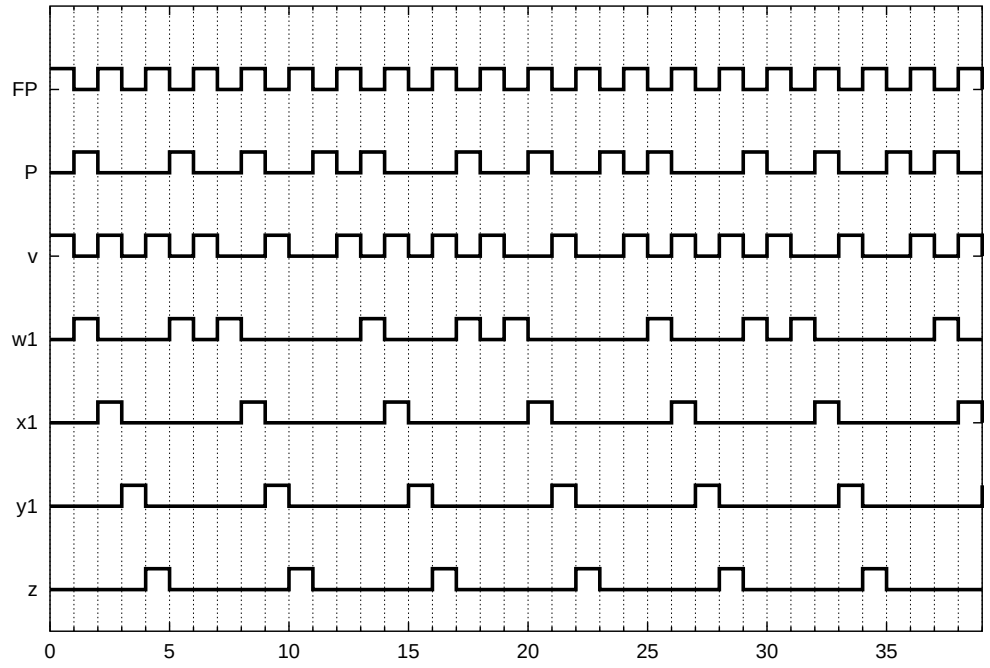


Figura 3.11: Comportamento do sistema com FP = Oscilador de período 2 (Modelo Binário).

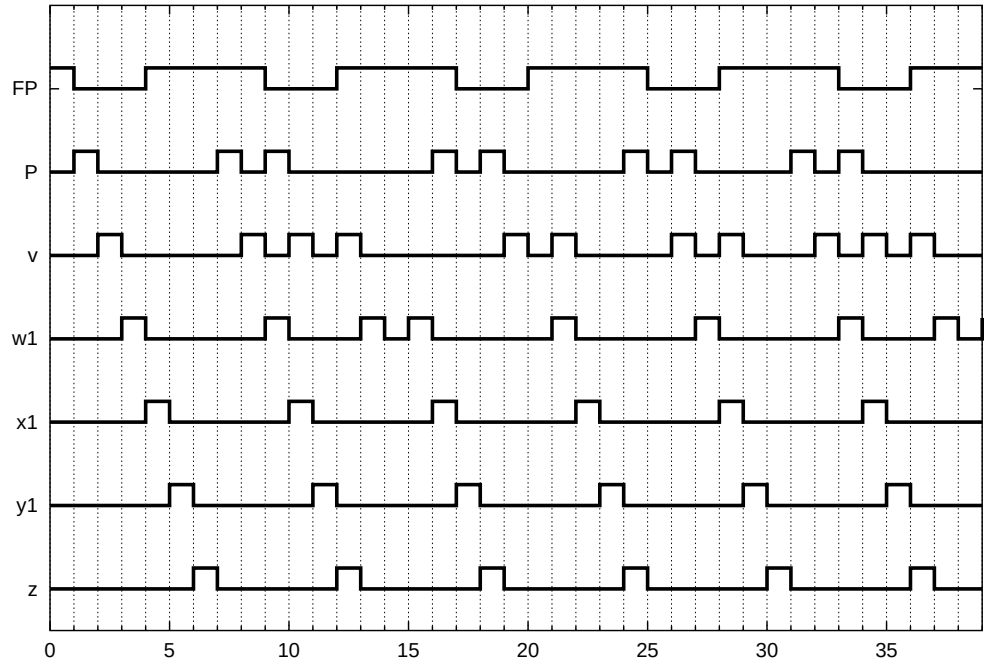
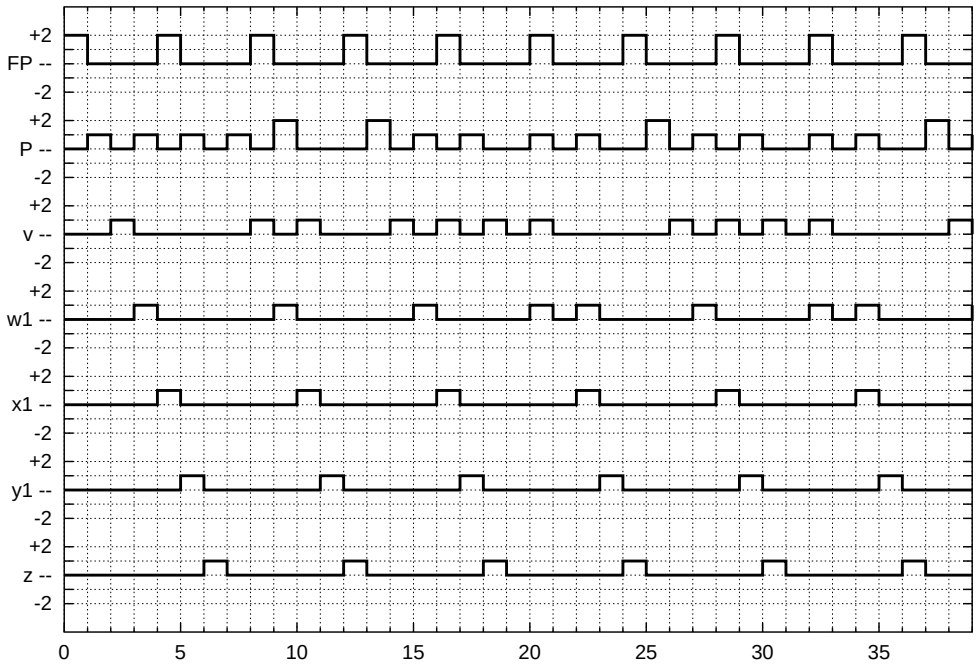


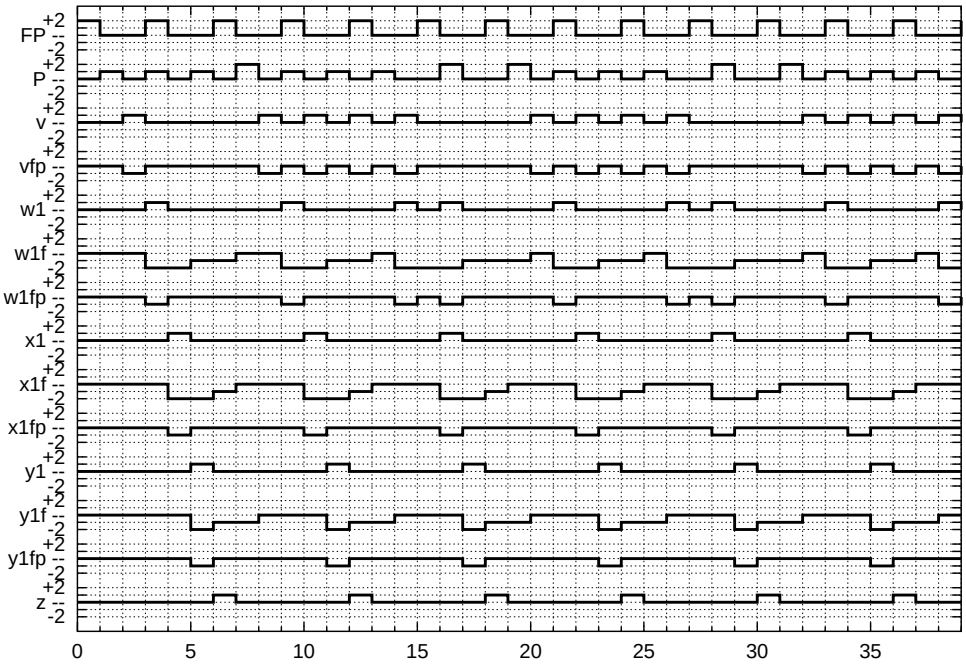
Figura 3.12: Comportamento do sistema com FP = Sinal periódico: "5 ligados, 3 desligados" (Modelo Binário).

(a) Sinais representativos a través do Ciclo Celular.

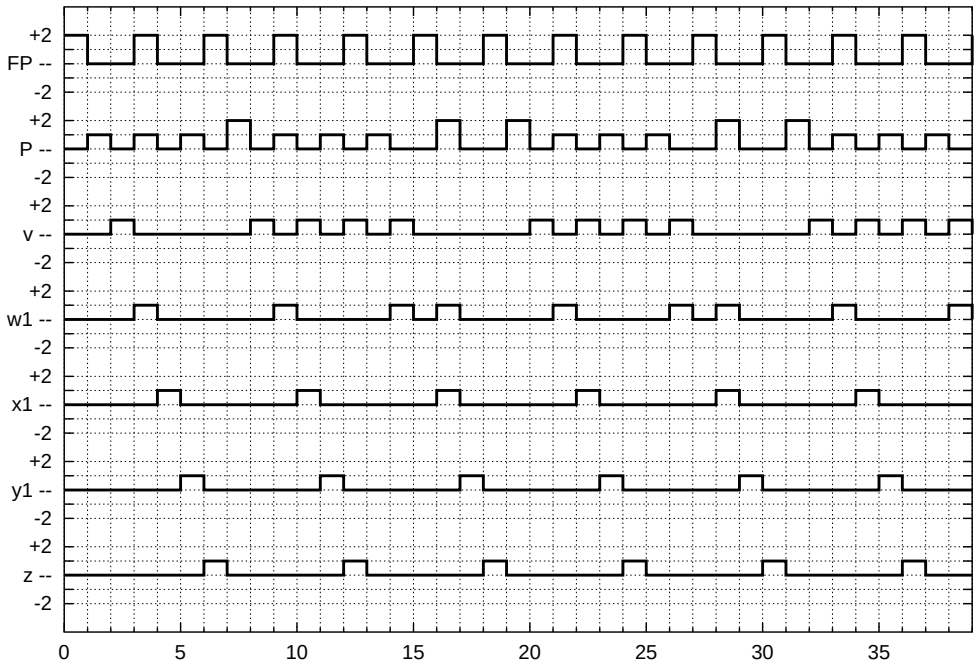


(b) Comportamento do sistema.

Figura 3.13: FP = Oscilador de período 4 (Modelo em Níveis de Cinza).



(a) Sinais representativos a través do Ciclo Celular.



(b) Comportamento do sistema.

Figura 3.14: FP = Oscilador de período 3 (Modelo em Níveis de Cinza).

Se o nível (e a freqüência) de expressão de P for suficientemente grande, os sinais excessivos conseguirão passar pelo ciclo celular, o qual deixará de ter o comportamento normal, (Figuras 3.15, 3.16, 3.17 e 3.18). Como conseqüência disso, a saída (final do ciclo ou expressão do gene z) não terá mais o mesmo período mínimo observado no comportamento normal.

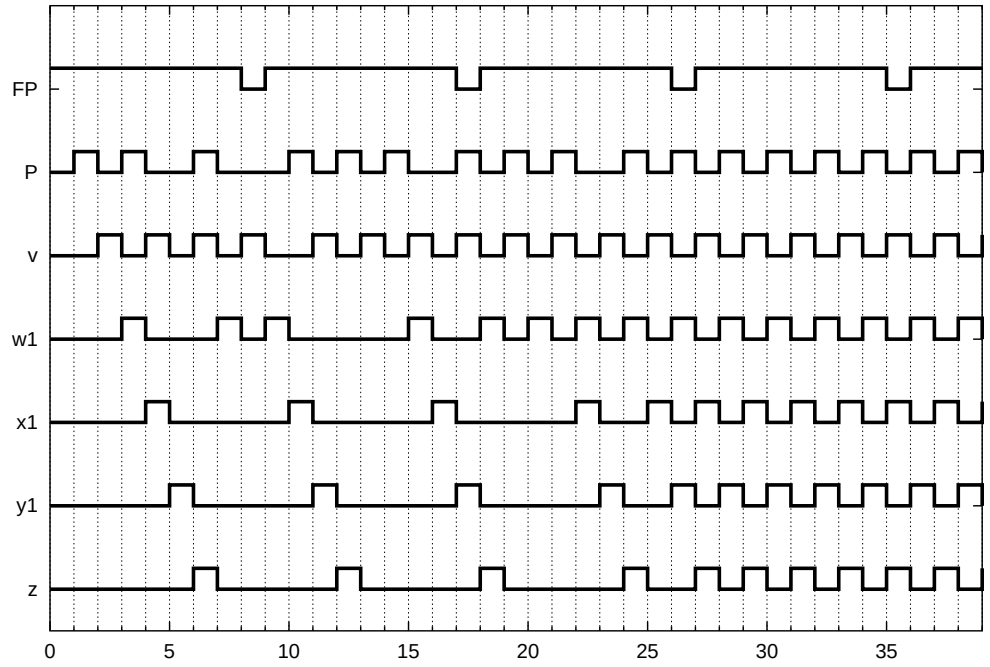
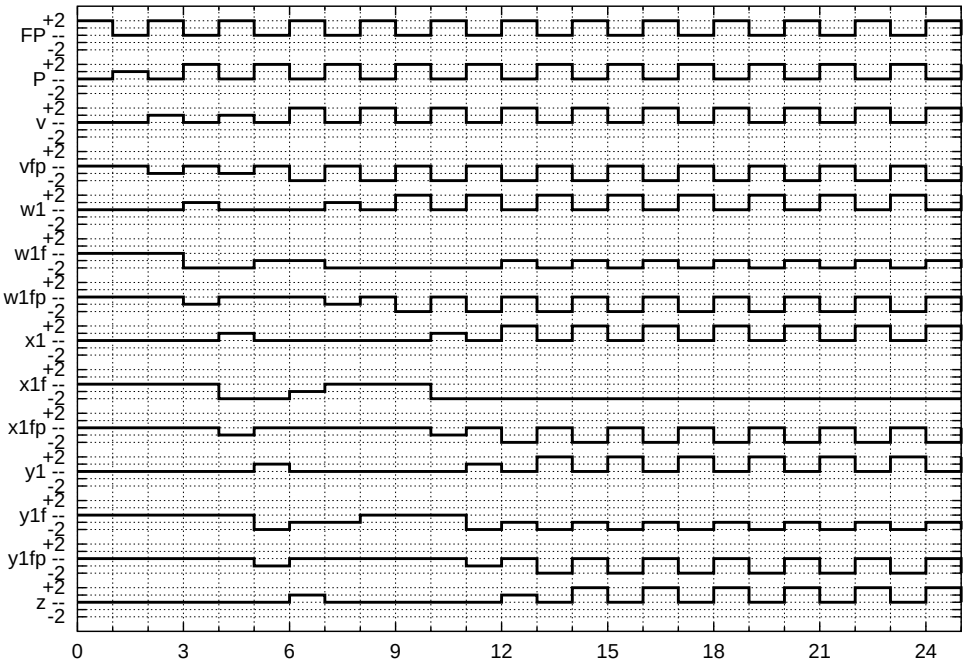
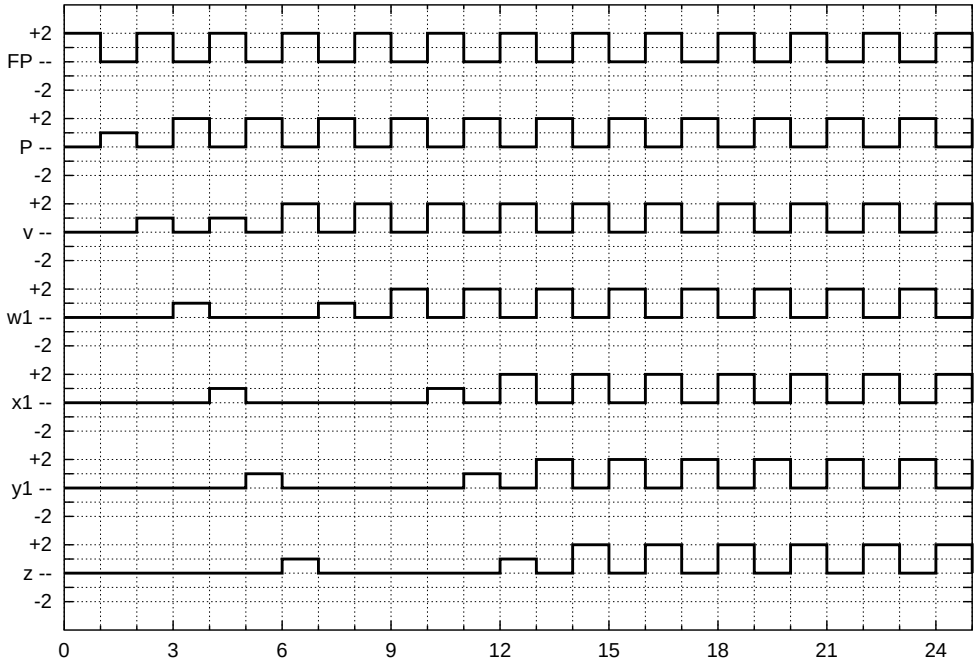


Figura 3.15: Comportamento do sistema com FP = Sinal periódico: "7 ligados, 1 desligado" (Modelo Binário).

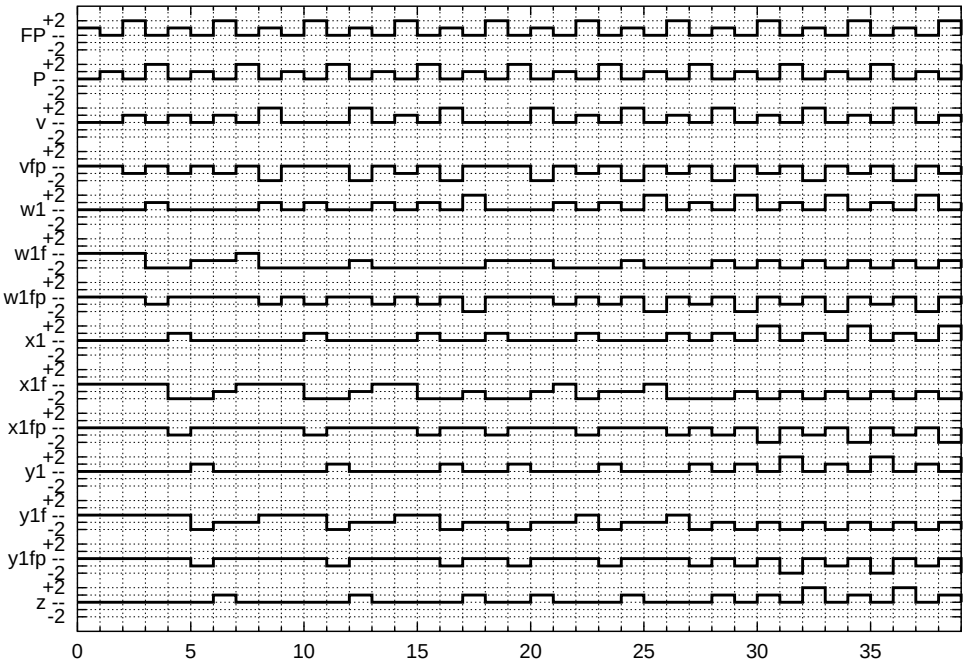


(a) Sinais representativos a través do Ciclo Celular.

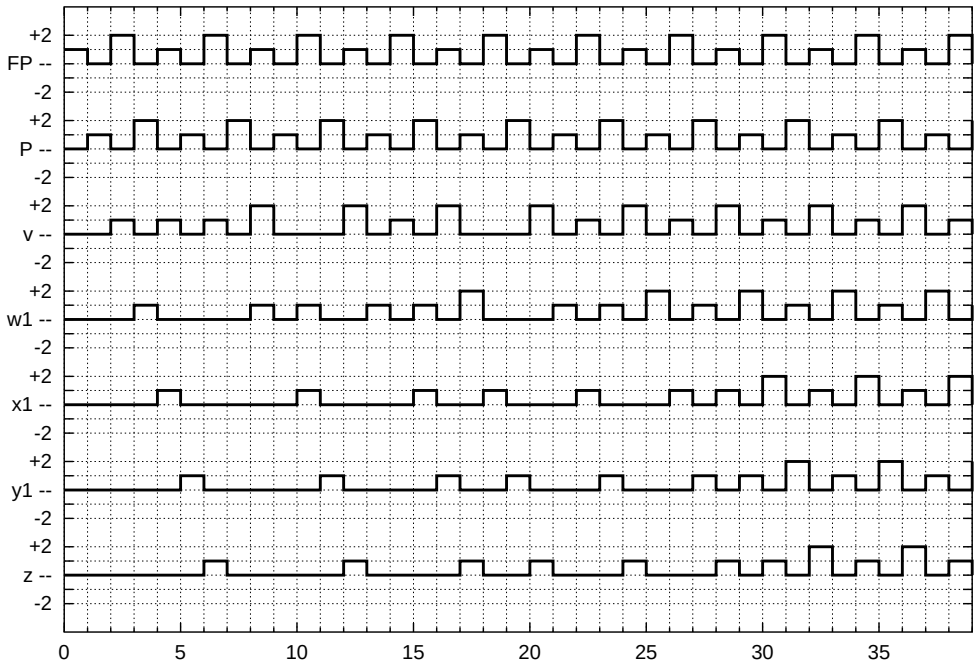


(b) Comportamento do sistema.

Figura 3.16: FP = Oscilador de período 2 (Modelo em Níveis de Cinza).

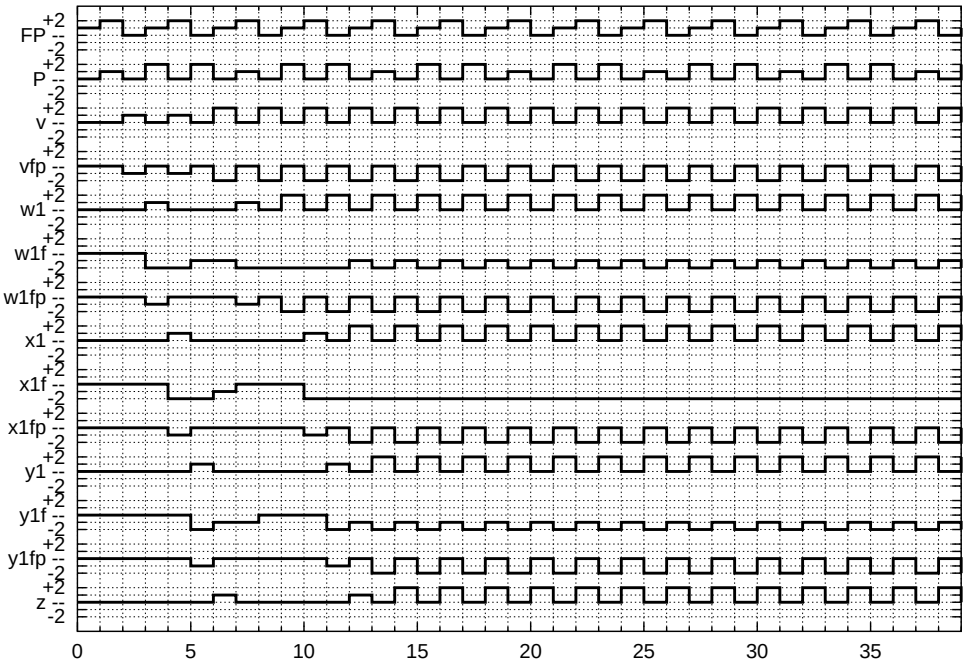


(a) Sinais representativos a través do Ciclo Celular.

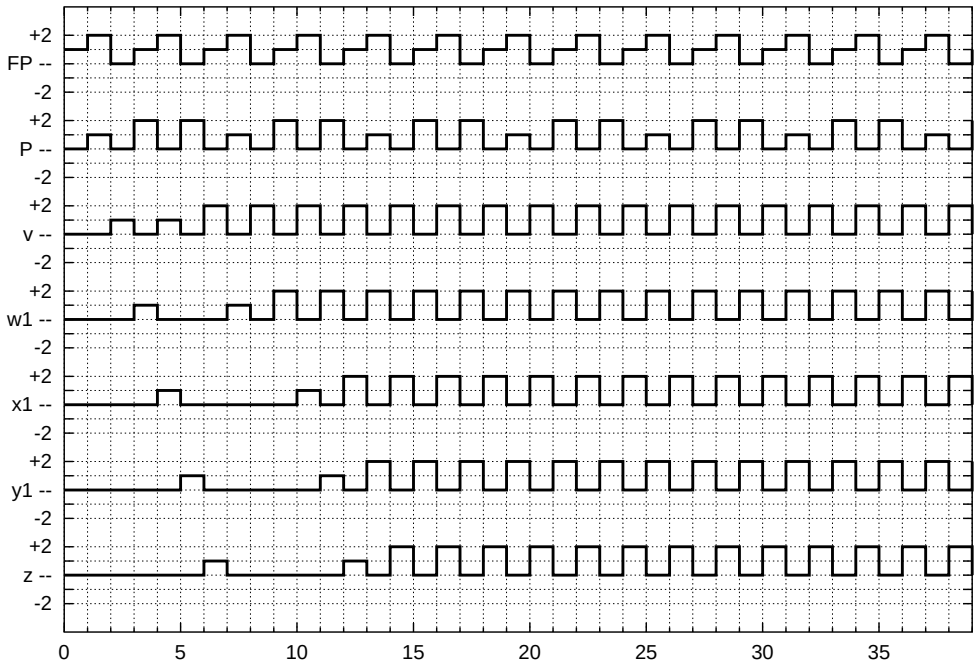


(b) Comportamento do sistema.

Figura 3.17: FP = Sinal Periódico com valores: 0,1,0,2 (Modelo em Níveis de Cinza).



(a) Sinais representativos a través do Ciclo Celular.

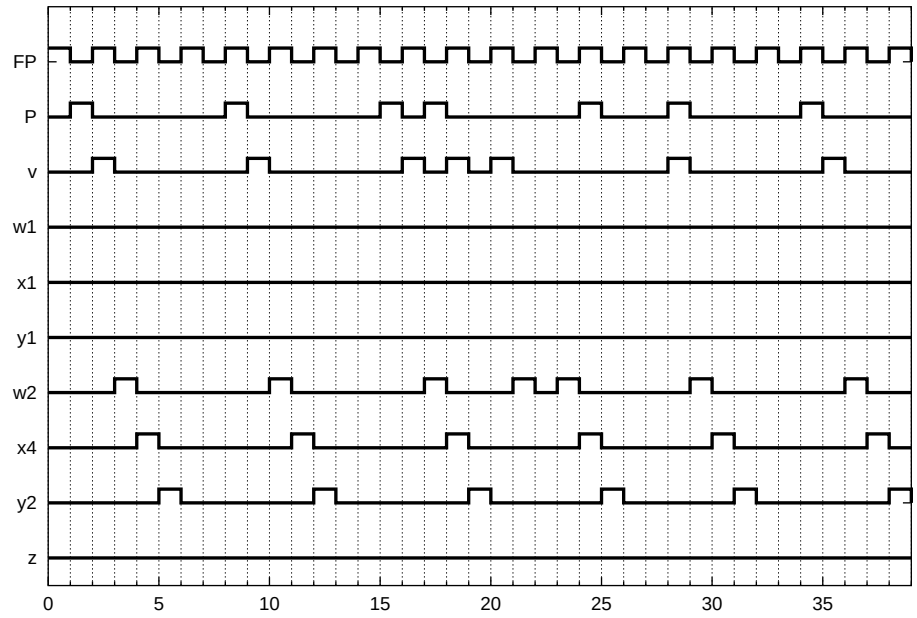


(b) Comportamento do sistema.

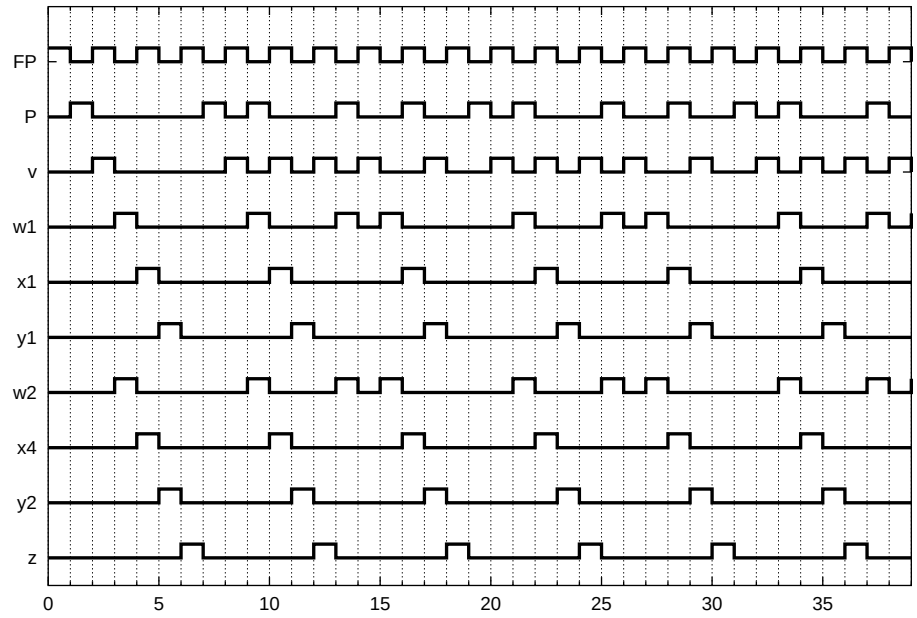
Figura 3.18: FP = Sinal Periódico com valores: 0,1,2 (Modelo em Níveis de Cinza).

Nas simulações das Figuras 3.19, 3.20, 3.21 e 3.22, foi testado o efeito de forçar um nível de expressão zero no gene $w1$ (efeito 'knock out'). Esta experiência é feita na prática para inferir a influência de um gene nos outros genes de uma rede de expressão gênica por comparação desta situação com as condições normais de expressão.

Nestes experimentos esperava-se que a inibição de um gene tivesse influência na ativação dos genes posteriores no caminho do sinal que estava se propagando. Este efeito foi observado, mas também observou-se que o nível de expressão dos genes no outro caminho do sinal também foi alterado em menor medida (Figuras 3.19 e 3.21). Isto acontece devido ao fato de que as realimentações para o gene P provenientes do gene inibido e os posteriores a ele no caminho do sinal não estão presentes (Figuras 3.20 e 3.22), o que muda o padrão de expressão do gene P e, portanto, de todos os genes na rede que são afetados por ele.

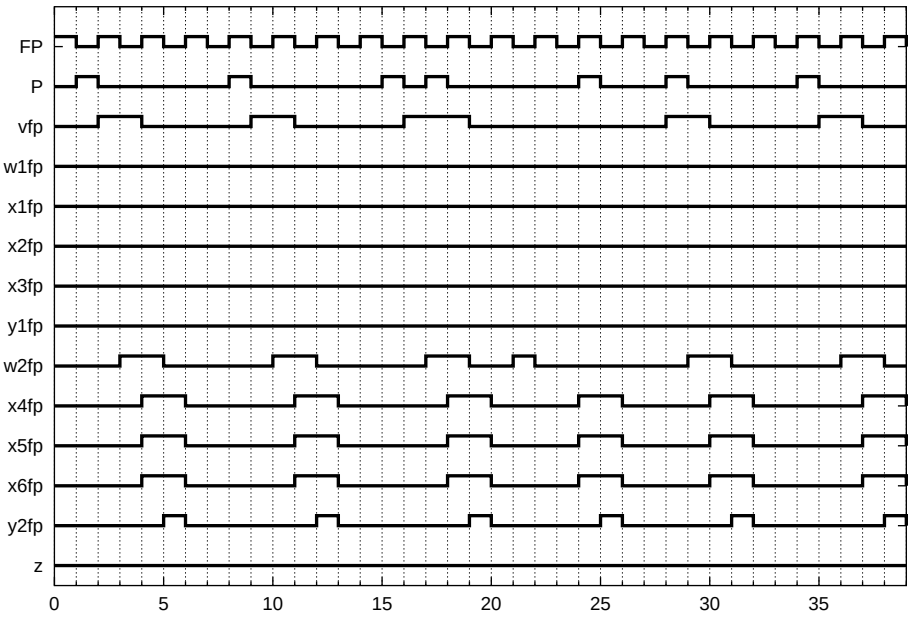


(a) Comportamento do sistema com w_1 "knock out".

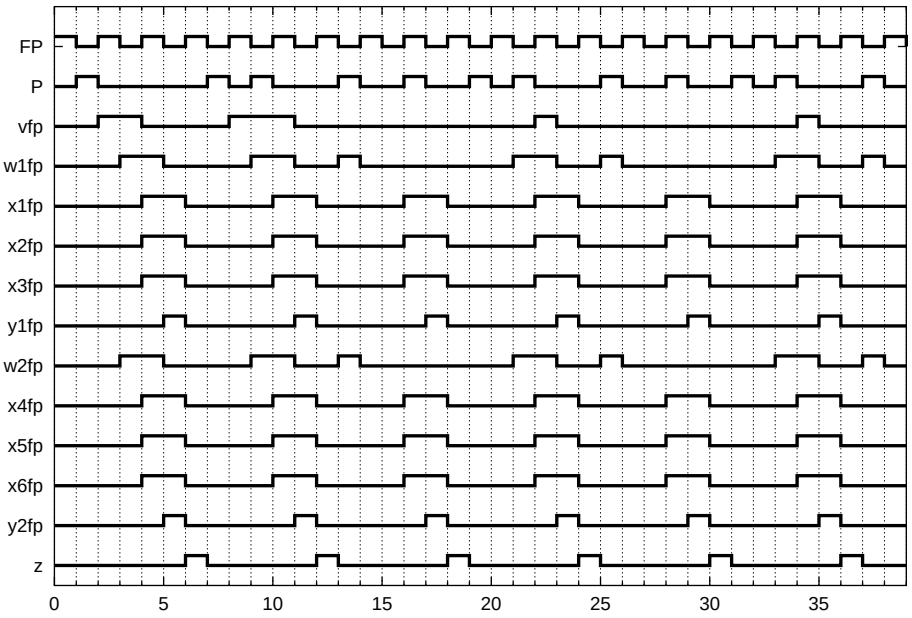


(b) Comportamento do sistema normal.

Figura 3.19: FP = Oscilador de período 2: w_1 "knock out" e sistema normal (Modelo Binário).

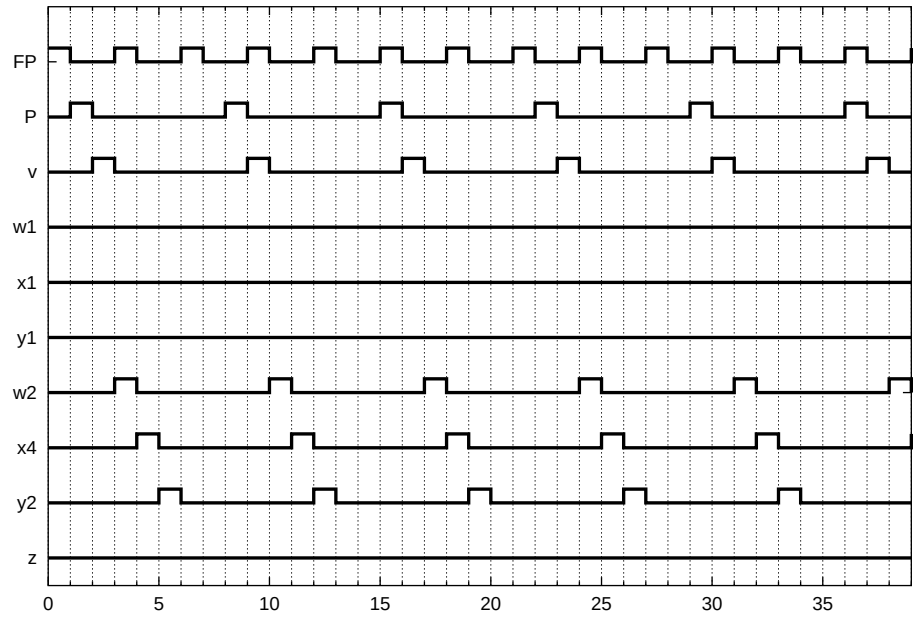


(a) Realimentações para o gene P com w_1 "knock out".

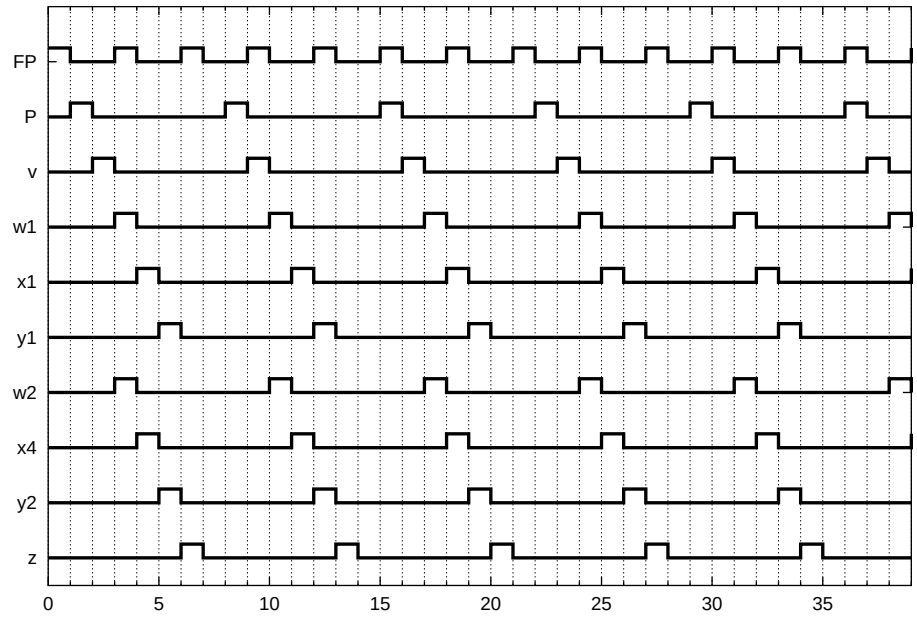


(b) Realimentações para o gene P do sistema normal.

Figura 3.20: FP = Oscilador de período 2: Realimentações para o gene P com w_1 "knock out" e no sistema normal (Modelo Binário).

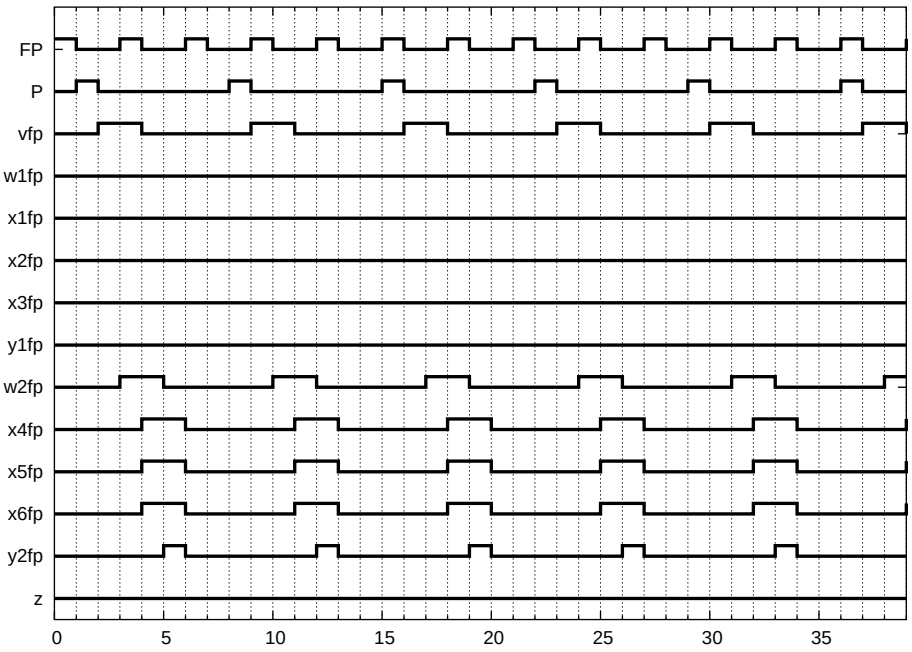


(a) Comportamento do sistema com w_1 "knock out".

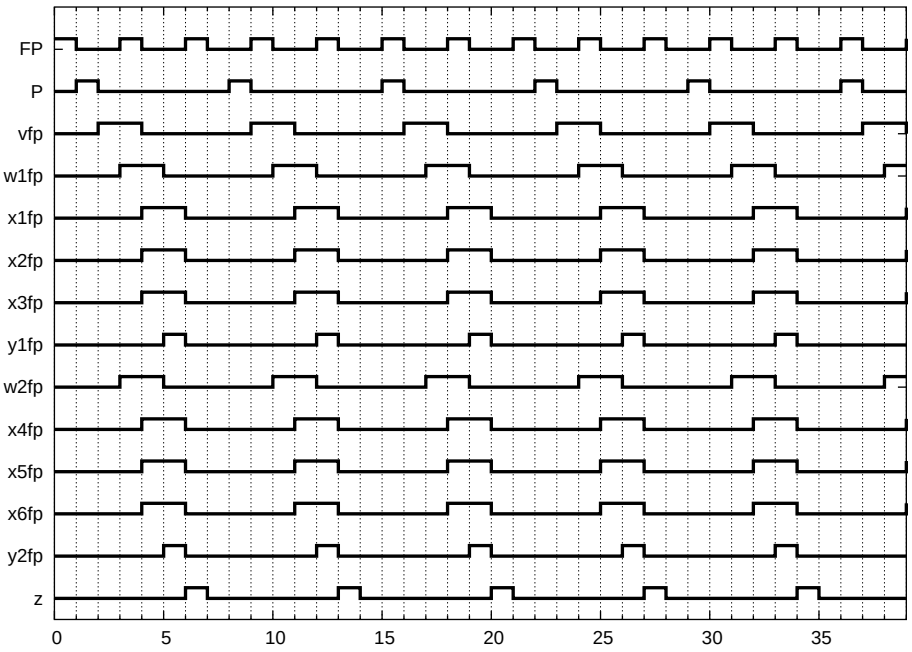


(b) Comportamento do sistema normal.

Figura 3.21: FP = Oscilador de período 3: w_1 "knock out" e sistema normal (Modelo Binário).



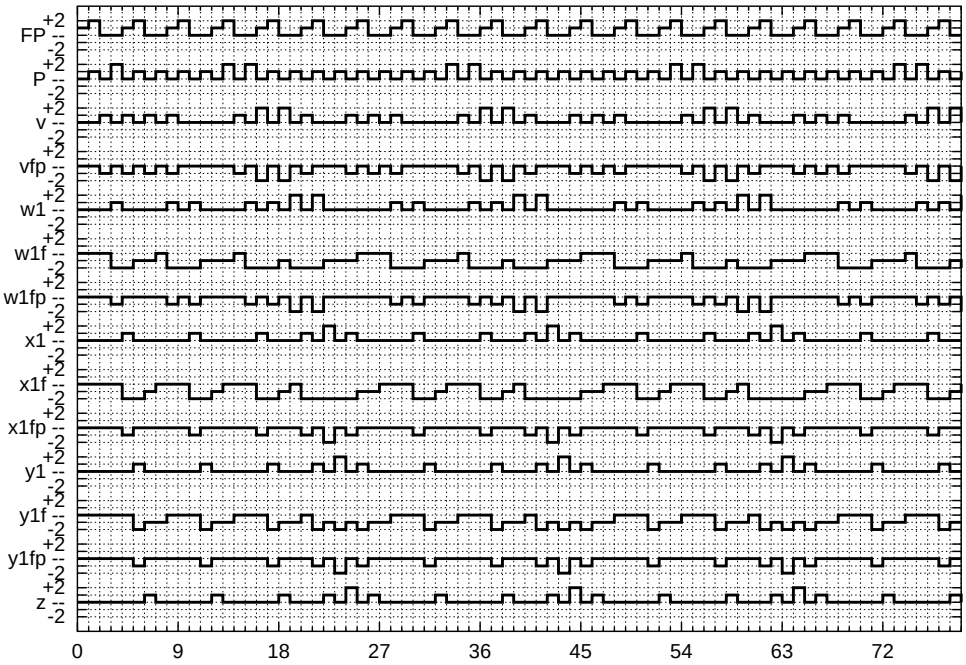
(a) Realimentações para o gene P com w_1 "knock out".



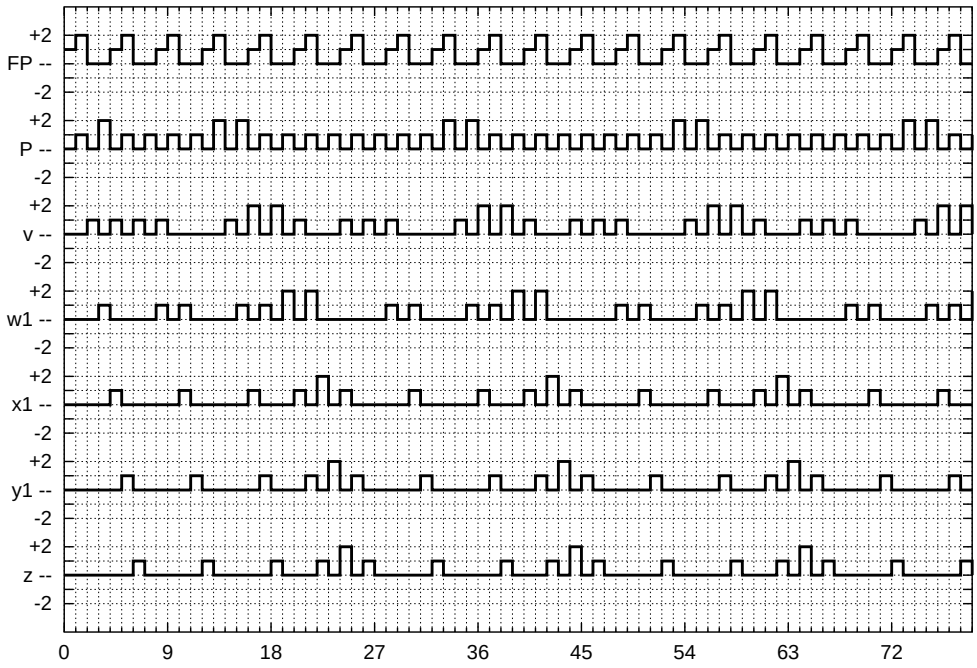
(b) Realimentações para o gene P do sistema normal.

Figura 3.22: FP = Oscilador de período 3: Realimentações para o gene P com w_1 "knock out" e no sistema normal (Modelo Binário).

Um exemplo interessante do funcionamento das realimentações pode ser visto nas simulações das Figuras 3.23 e 3.24. Um sinal FP forte quase consegue vencer a resistência do sistema, mas nesse momento atuam as realimentações para as camadas anteriores de genes e para o gene P (Apêndice B), fazendo com que o sinal que está se propagando seja atenuado. Ao diminuir este sinal, a força das realimentações também diminui, porém como o sinal FP continua sendo forte, o sistema volta à condição anterior, mostrando um comportamento cíclico.

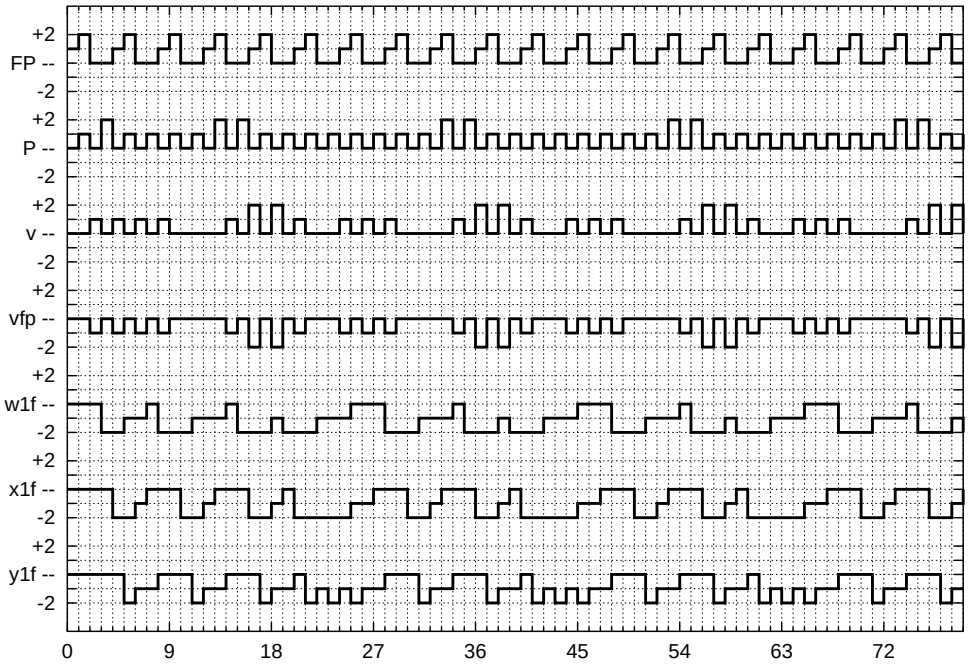


(a) Sinais representativos a través do Ciclo Celular.

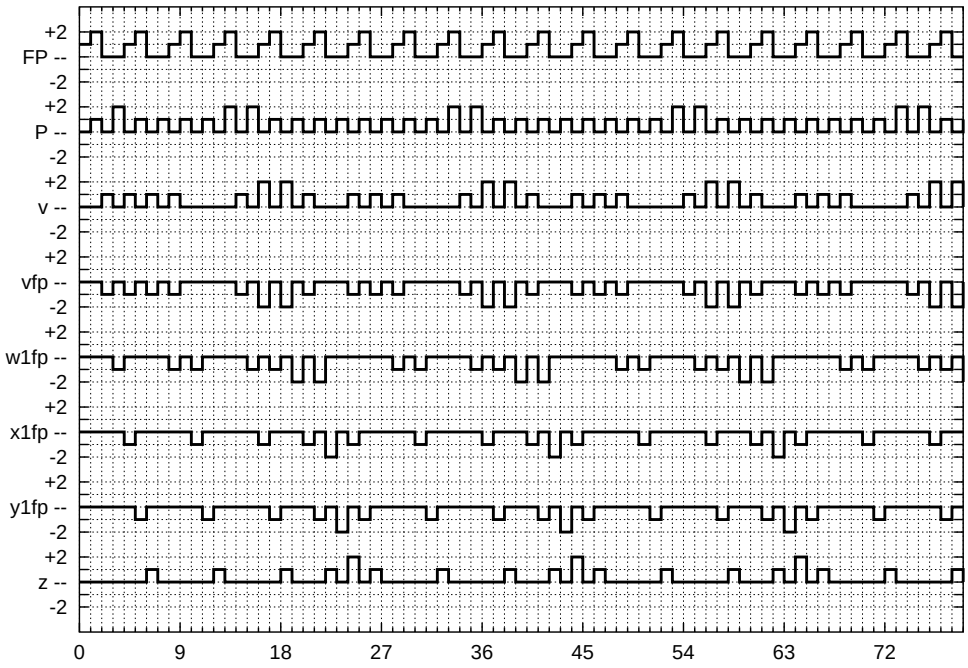


(b) Comportamento do sistema.

Figura 3.23: FP = Sinal Periódico com valores: 0,1,2,0 (Modelo em Níveis de Cinza).



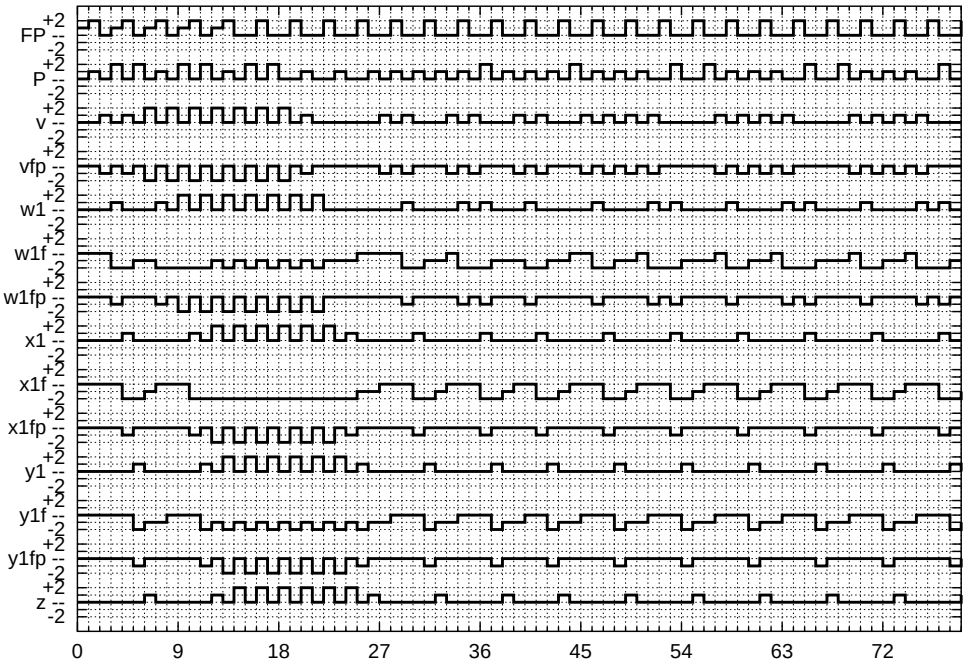
(a) Realimentações para a camada anterior.



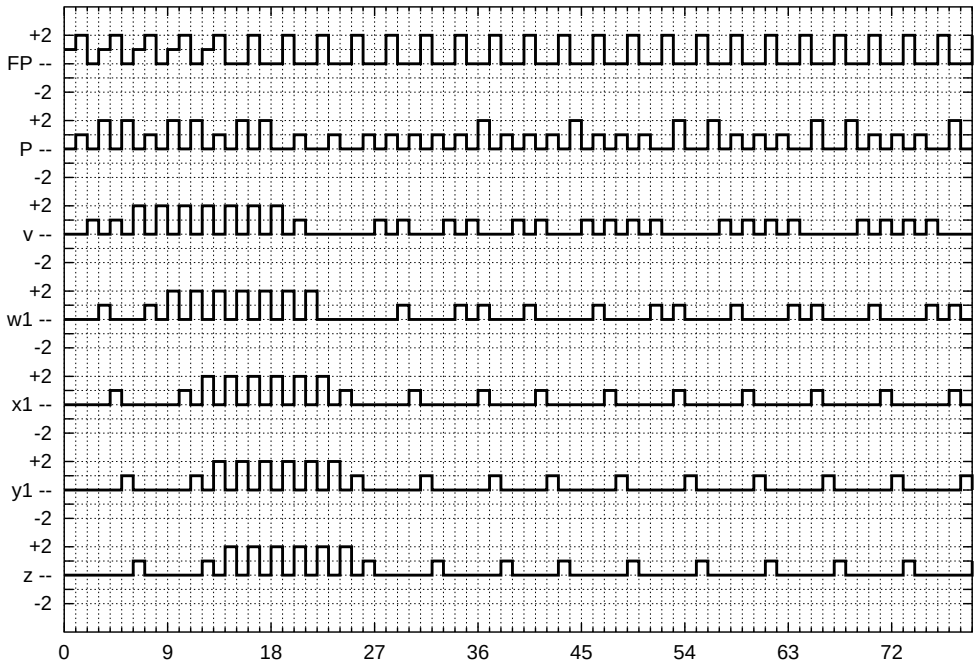
(b) Realimentações para o Gene P.

Figura 3.24: FP = Sinal Periódico com valores: 0,1,2,0 (Modelo em Níveis de Cinza).

Nas simulações da Figura 3.25 mostra-se o comportamento do sistema quando muda o sinal periódico presente em FP. Um sinal periódico FP suficientemente forte consegue vencer a resistência do sistema, mas depois o sinal de entrada muda para outro de intensidade apenas menor. O sistema adapta-se voltando ao funcionamento habitual para esse novo sinal FP, conseguindo resistir melhor ao excesso de excitação e apresentando um comportamento mais próximo do normal (veja o comportamento do gene z).



(a) Sinais representativos a través do Ciclo Celular.



(b) Comportamento do sistema.

Figura 3.25: FP = Sinal Periódico Variável (Modelo em Níveis de Cinza).

As Figuras 3.26 e 3.27 mostram um sinal FP obtido a partir de um sinal aleatório I e níveis de substância também aleatórios na primeira camada de genes u.

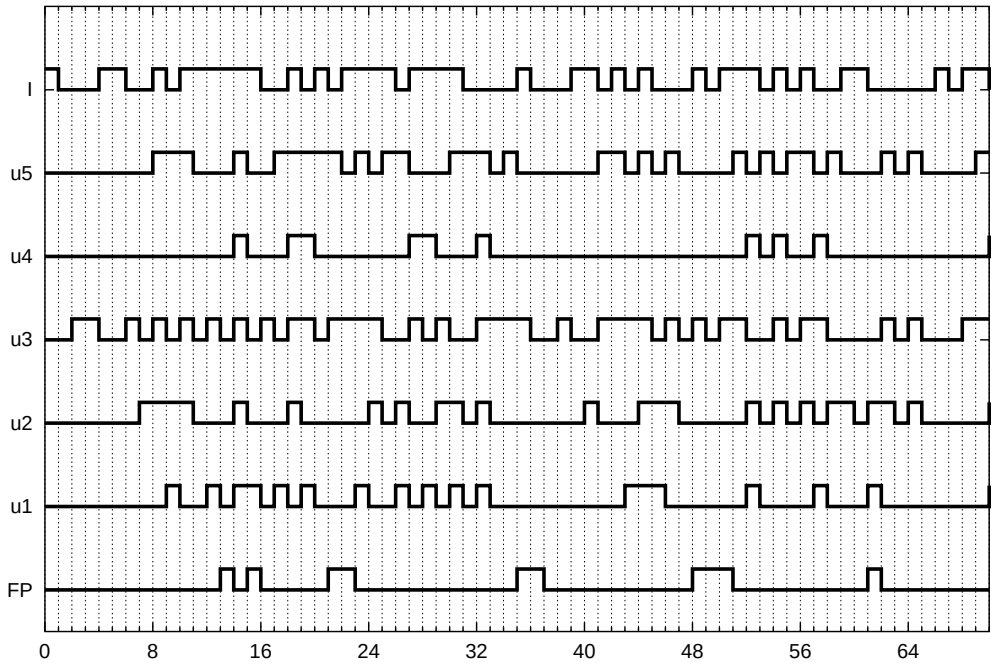


Figura 3.26: Sinal FP obtido a partir de uma sinal aleatório I (Modelo Binário).

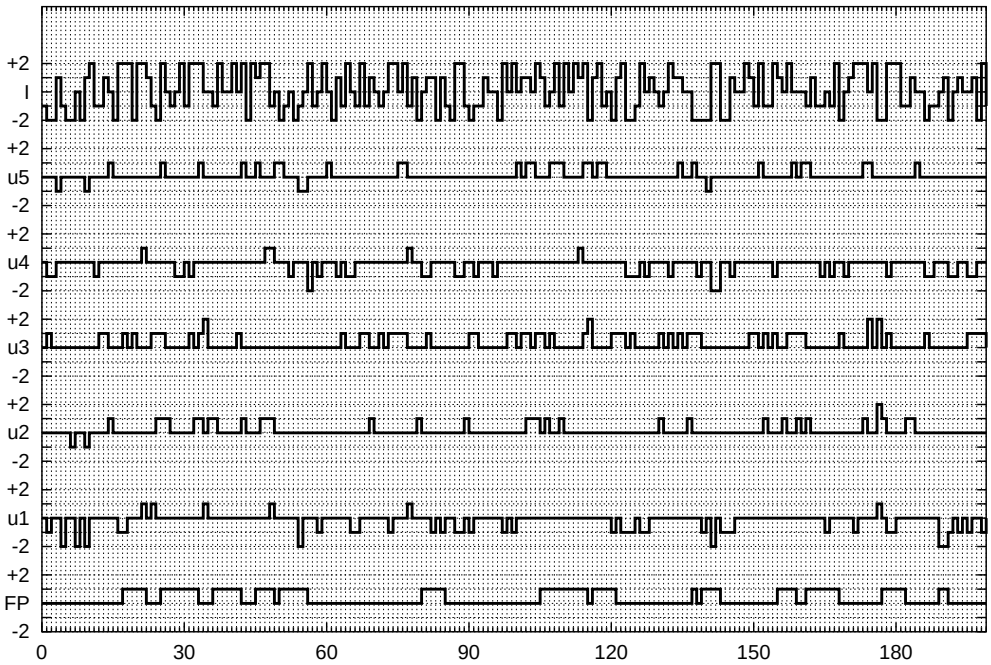


Figura 3.27: Sinal FP obtida a partir de uma entrada aleatória I (Modelo em Níveis de Cinza).

As Figuras 3.28 e 3.29 exemplificam o funcionamento do sistema completo a partir dos mesmos sinais aleatórios.

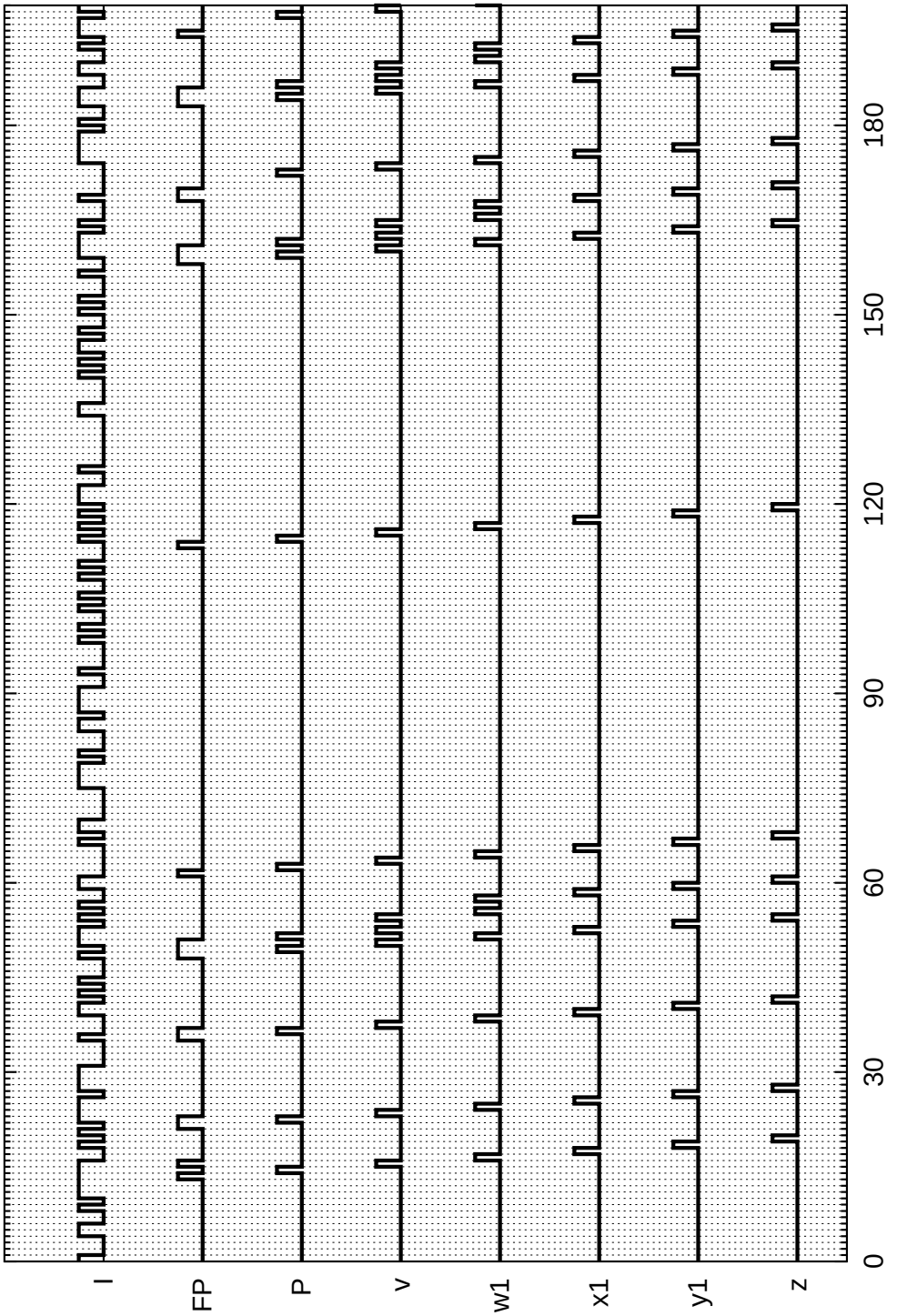
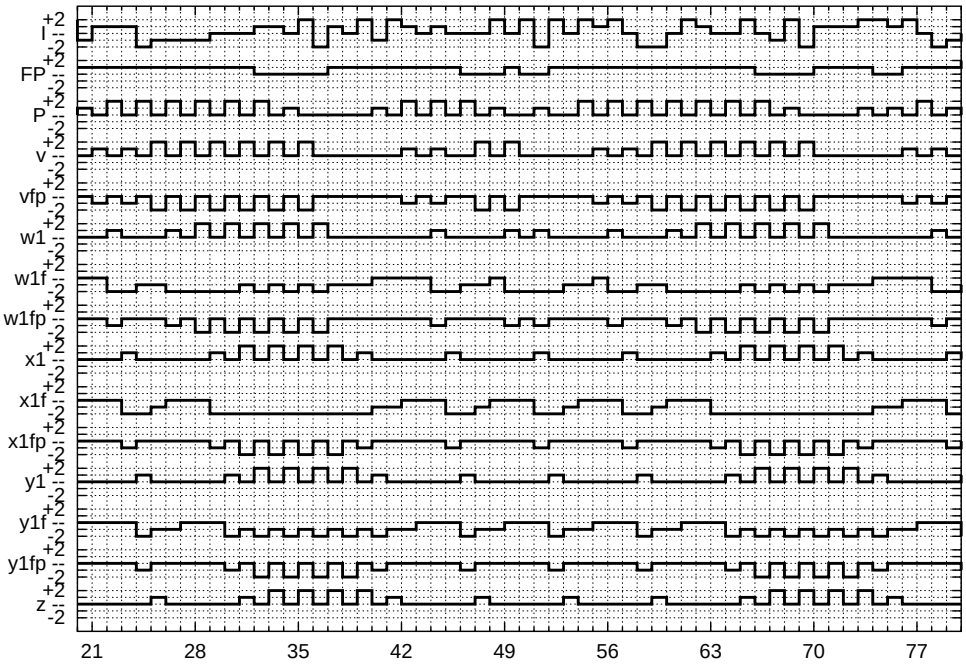
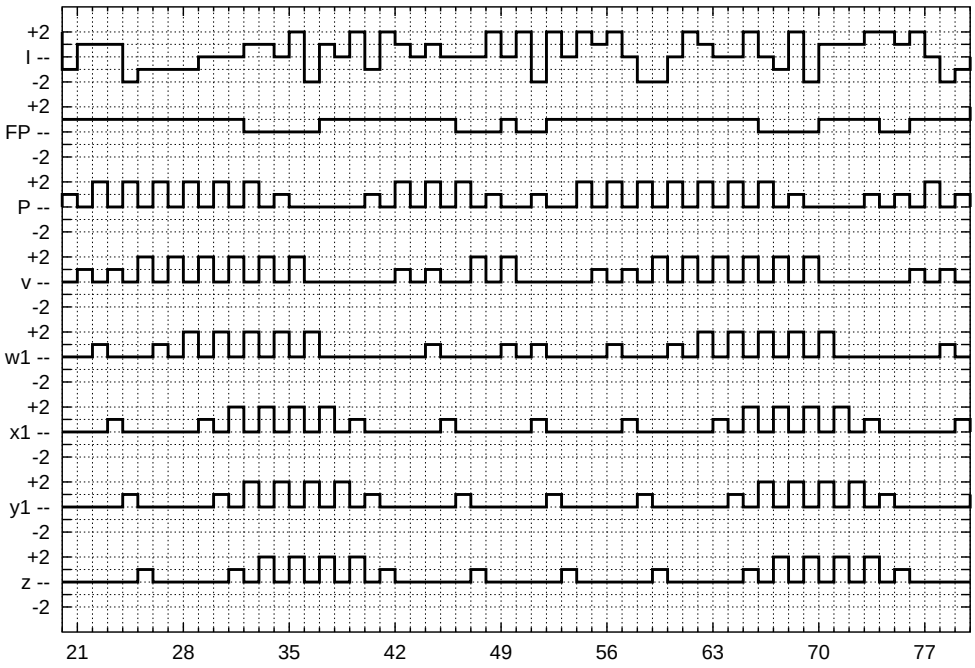


Figura 3.28: Comportamento do sistema completo exitado com um sinal aleatório I (Modelo Binário).



(a) Sinais representativos a través do Ciclo Celular.



(b) Comportamento do sistema.

Figura 3.29: Comportamento do sistema completo a partir de um sinal de entrada aleatório I (Modelo em Níveis de Cinza).

Capítulo 4

Identificação de Sistemas Livres

4.1 Aprendizado de Funções

A Teoria de Aprendizado Computacional estabelece métodos e fundamentos teóricos para os processos de aprendizado. O aprendizado de funções é utilizado na identificação de FLDSs para obter as componentes da função de transição, que determina as transições de estado, na forma das bases dos operadores de reticulados .

Qualquer processo capaz de "aprender um conceito", a partir de exemplos que o ilustram, é denominado de "aprendizado". O modelo básico de Valiant consiste no aprendizado de "conceitos" que podem ser modelados como funções Booleanas.

Consideremos um domínio D cujos elementos são as representações codificadas do "mundo real" e um conjunto Σ para descrever elementos de D denominado *alfabeto*. Σ_n denota o conjunto de todas as n-uplas de elementos de Σ.

Seja $X \subseteq \Sigma^n$. Um *conceito* c é uma função definida por $c : X \to \{0, 1\}$. $X \times \{0, 1\}$ é denominado *espaço de exemplos*. Um exemplo é um par (x, b) onde $x \in X$ e $b \in \{0, 1\}$. Se $b = 1$ indica que $x \in c$, se $b = 0$, então $x \notin c$. Uma amostra s de treinamento de tamanho

m é uma seqüência de m exemplos. Dizemos que s é consistente se $x_i = x_j \Rightarrow b_i = b_j$, $1 \leq i, j \leq m$. O *espaço de hipóteses* H é o espaço de todos os conceitos que podem ser aprendidos. O conceito $t \in H$ a ser aprendido é denotado *conceito alvo*.

Um *algoritmo de aprendizado* é uma função L que associa a cada amostra de treinamento, relativo a um conceito alvo $t \in H$, uma hipótese $h = L(s) \in H$.

Uma hipótese $h \in H$ é consistente com uma amostra s se $h(x_i) = b_i$ para cada $1 \leq i \leq m$. Um algoritmo de aprendizado é *consistente* se ele é consistente com todas as possíveis amostras de treinamento s consistentes.

Seja μ a distribuição de probabilidades associada a X, o modelo de aprendizado PAC [14] define o erro de uma hipótese $h \in H$ relativamente a um conceito alvo $t \in H$ como

$$er_\mu(h,t) = \mu\{x \in X : h(x) \neq t(x)\}$$

Seja X^m o espaço de amostras de treinamento de tamanho m, e dado $Y \subset X^m$, o valor $\mu^m(Y)$ representa a probabilidade de uma amostra aleatória de m exemplos, obtidos de X, segundo uma distribuição μ, pertencer a Y.

Um algoritmo L é PAC para um espaço de hipóteses H se, dados um número real δ, $(0 < \delta < 1)$, e um número real ε, $(0 < \varepsilon < 1)$, então existe um inteiro positivo $m_0 = m_0(\delta, \varepsilon)$ tal que para qualquer conceito alvo $t \in H$, e para qualquer distribuição de probabilidade μ sobre X, sempre que $m \geq m_0$, $\mu^m\{s \in S(m,t) : er_\mu(L(s), t < \varepsilon)\} > 1 - \delta$. Um resultado importante é que qualquer algoritmo consistente num espaço finito de hipóteses é PAC.

O modelo PAC pode ser estendido para incluir as situações em que existem exemplos

contraditórios [16]. Nesse caso no par (x, b), o elemento b não é determinístico, sendo gerado segundo uma distribuição condicional de probabilidade $p(b|c)$, que juntamente com uma distribuição μ sobre X, determina uma distribuição de probabilidade conjunta P sobre $X \times \{0, 1\}$. Os exemplos (x, b) são gerados segundo uma distribuição de probabilidade P.

O erro é caracterizado através de uma função $l_h : X \times \{0, 1\} \rightarrow [0, 1]$, denominada função de perda que "mede" a perda cometida ao se escolher uma determinada decisão h. O risco de uma hipótese $h \in H$, relativo à distribuição de probabilidade conjunta P sobre $X \times \{0, 1\}$ e à *função de perda* l_h, é definido por

$$r_P(l_h) = E[l_h(x, b)]$$

A hipótese de *menor risco* é aquela que minimiza a função de perda $r_P(l_h)$:

$$r_P(l_{h^*}) \leq r_P(l_h), \forall h \in H$$

Com base nesses conceitos, uma definição mais genérica de algoritmo PAC é a seguinte:

Um algoritmo L é PAC para um espaço de decisões H se, dados um número real δ, $(0 < \delta < 1)$, e um número real ε, $(0 < \varepsilon < 1)$, então existe um inteiro positivo $m_0 = m_0(\delta, \varepsilon)$ tal que para qualquer espaço de exemplos $X \times \{0, 1\}$, e para qualquer distribuição de probabilidade conjunta P sobre $X \times \{0, 1\}$, sempre que $m \geq m_0$, $P_m(|r_P(l_{L(s)}) - r_P(l_{h^*})| < \varepsilon) > 1 - \delta$.

Uma técnica que pode resultar conveniente para a melhora da precisão do aprendizado é a aplicação de restrições que reduzem o espaço das funções a serem aprendidas, ou

seja, reduzem o tamanho do espaço de hipóteses possíveis a um subconjunto $H' \subset H$. As restrições estão baseadas em conhecimento prévio que se tem sobre a função a ser aprendida, portanto, uma restrição será benéfica dependendo de quão boa seja a hipótese prévia acerca da função. Se a restrição é benéfica, significa que para cumprir a condição PAC dado um par (δ, ε) será requerido um tamanho mínimo de amostra m_0 menor, ou equivalentemente, para um tamanho de amostra m_0 dado, o erro cometido será menor. A aplicação da restrição, baseada no conhecimento prévio que se tem da função, faz que a observação de um dado permita inferir informação referente a outros dados não observados. Esto tem um efeito equivalente à amplificação dos dados disponíveis e como conseqüência uma melhora da qualidade da identificação realizada.

Exemplos de restrições:

A maneira de exemplo, no caso do processamento de imagens, são utilizados mapeamentos do seguinte tipo. Seja $E = Z^2$ o plano inteiro e $\mathcal{P}(E)$ o conjunto de partes de E. Considere o mapeamento $\Psi : \mathcal{P}(E) \to \mathcal{P}(E)$. As seguintes propriedades definem restrições no espaço de funções a serem aprendidas [21, 22, 23]:

- Uma função Ψ é *crescente* se e somente se, para todo conjunto A e B,

$$A \subset B \Rightarrow \Psi(A) \subset \Psi(B)$$

- Uma função Ψ é *decrescente* se e somente se, para todo conjunto A e B,

$$A \subset B \Rightarrow \Psi(B) \subset \Psi(A)$$

- Uma função Ψ é *idempotente* se e somente se, para todo conjunto A,

$$\Psi(\Psi(A)) = \Psi(A)$$

- Uma função Ψ é *extensiva* se e somente se, para todo conjunto A,

$$A \subset \Psi(A)$$

- Uma função Ψ é *anti-extensiva* se e somente se, para todo conjunto A,

$$\Psi(A) \subset A$$

4.1.1 Algoritmos de Aprendizado de Funções

Um algoritmo de aprendizado de funções Booleanas PAC trivial pode ser escrito da seguinte forma [12]:

```
Entrada:   - conjunto de exemplos S = {(x_i, b_i) : 1 ≤ i ≤ m}
Saida:     - uma hipotese (função Booleana) f

  {
      f = ∅              % conjunto vazio
      Para i de 1 ate m faça
          se b_i = 1 então faça
              f = f ∪ {x_i};
      Retorne f;
  }
```

Ele fornece como resultado uma função Booleana na forma de soma de produtos canônicos (FND). Na prática essa forma de representar uma função Booleana não é interessante pois o número de termos e variáveis envolvidos é grande, tornando difícil a sua manipulação. Procura-se então encontrar uma expressão equivalente mais simples, que pode ser obtida por exemplo através de simplificação de expressões. As funções Booleanas na forma FND podem ser minimizadas pelo algoritmo de Quine-McCluskey [9, 10], mas quando o número de variáveis envolvidas é grande, este algoritmo torna-se inviável pois requer grande espaço de memória e tempo de processamento. O algoritmo ISI ("Incremental Splitting of Intervals") baseado no "Particionamento Sucessivo de Intervalos" é muito mais eficiente neste sentido [12].

Algoritmo ISI

Os mintermos para os quais a função toma o valor 1 são denominados *mintermos positivos*, e aqueles para os quais ela toma valor 0 são denominados *mintermos negativos*.

O processo é inicializado com o n-cubo (função Booleana constante igual a 1) e a partir dele os pontos correspondentes aos mintermos negativos de f são eliminados sucessivamente, conservando só os intervalos que sejam maximais, ou seja que não estão contidos em outros de dimensão maior. Ao final da extração de todos os mintermos negativos, todos os pontos restantes do n-cubo, representados em termos de um conjunto de intervalos maximais, correspondem ao conjunto de mintermos positivos e mais aqueles que não foram definidos ("don't care"). Como reticulado Booleano o n-cubo pode ser visto como um intervalo $[\emptyset, W]$, onde $|W| = n$. A representação minimal de f é obtida em termos de um conjunto de intervalos maximais. A característica fundamental deste algoritmo é que ele representa o conjunto de todos os pontos de um intervalo, a menos de um, em termos de um conjunto de intervalos maximais. O seguinte teorema formaliza este importante resultado [12]:

Seja $[A, B] \subseteq P(W)$ e $C \in [A, B]$. *Então*

$$[A, B] - \{C\} = \{[A, B \cap \overline{\{a\}}] : a \in C\} \cup \{[A \cup \{b\}, B] : b \in \overline{C}\},$$

onde o complemento é em relação a W. *Os intervalos do lado direito da igualdade são maximais.*

Corolário: *Seja* $[A,B] \subseteq P(W)$ e $C \in [A,B]$. *Então*

$$[A,B] - \{C\} = \{[A, B \cap \overline{\{a\}}] : a \in C \cap \overline{A}\} \cup \{[\{b\} \cup A, B] : b \in \overline{C} \cap B\}$$

Um mintermo negativo P_1 é extraído do intervalo inicial $[\emptyset, W]$, e o conjunto $[\emptyset, W] - P_1$ é expresso como um conjunto N de intervalos maximais. O segundo mintermo negativo P_2 deve ser extraído de cada um dos intervalos de $[\emptyset, W] - P_1$ que o contém. Para garantir que o conjunto de todos o intervalos é maximal, antes de extrair outro exemplo negativo devem-se eliminar os intervalos que estão contidos em outros de dimensão maior. Ao final da extração de todos os mintermos negativos, obtém-se o conjunto de intervalos maximais correspondente à representação minimal de f, ou seja todos os implicantes primos de f. O algoritmo ISI em pseudocódigo é o seguinte [12]:

```
Entrada:   - conjunto de mintermos positivos e negativos
           - intervalos de inicialização
Saida:     - conjunto de implicantes primos essenciais

  {
      T = intervalos de inicialização;
      Para cada mintermo negativo P faça:
          N = ∅;
          T0 = intervalos de T que contem P;
          T1 = intervalos de T que não contem P;
          Para cada intervalo I de T0 faça:
              Elimine P de I e gere os subintervalos I'i;
              Para cada I'i,
                  se I'i ∈ T1
                      então N = N ∪ {I'i}
          T = T1 ∪ N;
      Retorne T;
  }
```

Do conjunto de implicantes primos selecionam-se os essenciais.

O algoritmo ISI é extremamente eficiente quando o número de "don't cares" (mintermos não definidos) é grande, em particular ele é muito mais eficiente do que o algoritmo de Quine-McCluskey nestes casos.

O algoritmo ISI foi originalmente desenvolvido para simplificar funções com entradas e saídas binárias (ou seja, funções do tipo $f : \{0,1\}^n \rightarrow \{0,1\}$). Em seguida foi extendido para simplificar funções com entrada binária e saída multivalorada [24] (ou seja, funções do tipo $f : \{0,1\}^n \rightarrow \{0,1,\ldots,k\}$) e, mais recentemente, para simplificar a representação de funções em níveis de cinza com saída em níveis de cinza. Estas extensões são denominadas, respectivamente, de *ISI multiclassificação* e *ISI níveis de cinza*(*GISI*) [25].

Na prática o aprendizado das funções Booleanas é efetuado a partir de conjuntos de dados amostrados ou observados correspondentes a estados do sistema dinâmico. Dado que não se dispõe de informação sobre todos os estados possíveis, é necessária a utilização de técnicas estatísticas no aprendizado, o que implica na existência de um certo erro de estimação.

A aplicação de restrições sobre o espaço das funções a ser aprendidas, baseadas em algum conhecimento ou suposição adequada do sistema, reduz o espaço de possíveis funções, e pode ter como conseqüência positiva uma diminuição considerável do erro da estimação obtido, para uma mesma quantidade de dados de treinamento [26].

Os conjuntos de dados a serem manipulados sobre os quais se procuram métodos computacionais para maximizar a inferência funcional são muito grandes. Uma redução do número de dependências funcionais (variáveis de entrada das componentes da função de transição) que estão sendo consideradas, com a conseqüente redução das operações

computacionais, pode ser obtida analisando sistematicamente a informação mútua entre os estados de entrada e saída [27, 28]. Assim, é possível inferir os subconjuntos de variáveis que determinam mais fortemente o comportamento de cada variável de estado (nível de expressão de um gene da rede).

Com o objetivo de aprender o uso das técnicas de aprendizado computacional, participei do projeto de Árvores de Classificação por Multi-resolução no Projeto de OCRs [29]. O Apêndice C mostra o trabalho realizado, onde foi criada uma técnica de geração automática para a criação destas árvores de classificação, a qual pode ser aplicada para a obtenção de multiclassificadores especialmente adaptados a problemas específicos de classificação.

4.2 Identificação de Sistemas Dinâmicos Finitos

A identificação de um Sistema Dinâmico Finito ou FDS $S(\psi)$ consiste na aplicação de um algoritmo L, o *algoritmo de identificação*, que constrói uma boa aproximação $S(\phi)$ para $S(\psi)$ a partir de amostras da dinâmica do sistema. Para propósitos práticos, L constrói a função de transição ϕ numa representação algébrica conveniente. Quando $S(\phi)$ é um FLDS, usualmente L fornece a base da função de transição ϕ [2, 30].

Suponha que o espaço de estados X está estruturado com alguma distribuição de probabilidades μ, fixa mas desconhecida. Uma *amostra de treinamento* s para um FDS *alvo* $S(\psi)$ é gerada extraindo de X, de acordo com μ, as condições iniciais independentes e aleátorias $x_1, \ldots, x_m \in X$. Para cada uma das condições iniciais x_i, amostramos as k próximas iterações de ψ:

$$s(\psi, x_i) = \{x_i, \psi(x_i), \psi^{(2)}(x_i), \ldots, \psi^{(k)}(x_i)\}$$

Seja l uma *função de perda*, isto é, uma função de $X^k \times X^k$ em $[0, \infty)$ que mede a distância entre dinâmicas em X^k. A *função de perda de custo simples* l_{SC} é aquela que conta o número de diferenças correspondentes nas duas dinâmicas comparadas, isto é, para cada $u, v \in X^k$,

$$l_{SC}(u, v) = |\{u[t] : u[t] \neq v[t], t \in \{1, \ldots, k\}\}|,$$

onde $u[t]$ e $v[t]$ denotam, respectivamente, as componentes de u e v.

Supõe-se que a função de transição do FDS alvo ψ pertence a algum espaço de hipótese

$H \subseteq X^X$. Dada a função de transição do sistema alvo $\psi \in H$, o *erro* segundo l de qualquer hipótese $\phi \in H$ será

$$Er_\mu(\phi, \psi, k) = \sum_{x \in X} l(s(\psi, x), s(\phi, x)) P(s(\psi, x), s(\phi, x)),$$

onde $P(s(\psi, x), s(\phi, x))$ é a probabilidade conjunta de $s(\psi, x)$ e $s(\phi, x)$. Nesta formulação, com um sistema alvo determinístico, $P(s(\psi, x), s(\phi, x)) = \mu\{x\}$, o erro de custo simples é dado, equivalentemente por

$$Er_\mu(\phi, \psi, k) = \sum_{j=1}^{k} \mu\{x \in X : \phi^{(j)}(x) \neq \psi^{(j)}(x)\}$$

onde k pode variar de 1 até $|X|$.

Por simplicidade, se $k = |X|$, denotaremos o erro anterior por $Er_\mu(\phi, \psi)$. É claro que se $k_1 \leq k_2$ então $Er_\mu(\phi, \psi, k_1) \leq Er_\mu(\phi, \psi, k_2)$, e assim o aumento de k permite um refinamento na medição do erro.

Exemplo. Seja $X = \{1, 2, 3, 4, 5\}$ o espaço de estados, $\psi = (2, 3, 1, 5, 4)$ a função de transição do sistema alvo, ou seja, $\psi(1) = 2$, $\psi(2) = 3$, ..., $\psi(5) = 4$, e suponha que μ é uma distribuição uniforme sobre X. Se a função de transição do sistema hipótese é $\phi = (4, 3, 1, 5, 2)$, então temos:

$\psi = (2, 3, 1, 5, 4)$	$\phi = (4, 3, 1, 5, 2)$	$\mu\{x \ : \ \psi(x) \neq \phi(x)\} = 2/5$
$\psi^2 = (3, 1, 2, 4, 5)$	$\phi^2 = (5, 1, 4, 2, 3)$	$\mu\{x \ : \ \psi^2(x) \neq \phi^2(x)\} = 4/5$
$\psi^3 = (1, 2, 3, 5, 4)$	$\phi^3 = (2, 4, 5, 3, 1)$	$\mu\{x \ : \ \psi^3(x) \neq \phi^3(x)\} = 1$
$\psi^4 = (2, 3, 1, 4, 5)$	$\phi^4 = (3, 5, 2, 1, 4)$	$\mu\{x \ : \ \psi^4(x) \neq \phi^4(x)\} = 1$
$\psi^5 = (3, 1, 2, 5, 4)$	$\phi^5 = (1, 2, 3, 4, 5)$	$\mu\{x \ : \ \psi^5(x) \neq \phi^5(x)\} = 1$

e o erro é:

$$Er_{\mu}(\phi, \psi) = \frac{21}{5}.$$

Se a função de transição da hipótese é $\varphi = (3, 1, 2, 5, 4)$, temos:

$\psi = (2, 3, 1, 5, 4)$	$\varphi = (3, 1, 2, 5, 4)$	$\mu\{x \ : \ \psi(x) \neq \varphi(x)\} = 3/5$
$\psi^2 = (3, 1, 2, 4, 5)$	$\varphi^2 = (2, 3, 1, 4, 5)$	$\mu\{x \ : \ \psi^2(x) \neq \varphi^2(x)\} = 3/5$
$\psi^3 = (1, 2, 3, 5, 4)$	$\varphi^3 = (1, 2, 3, 5, 4)$	$\mu\{x \ : \ \psi^3(x) \neq \varphi^3(x)\} = 0$
$\psi^4 = (2, 3, 1, 4, 5)$	$\varphi^4 = (3, 1, 2, 4, 5)$	$\mu\{x \ : \ \psi^4(x) \neq \varphi^4(x)\} = 3/5$
$\psi^5 = (3, 1, 2, 5, 4)$	$\varphi^5 = (2, 3, 1, 5, 4)$	$\mu\{x \ : \ \psi^5(x) \neq \varphi^5(x)\} = 3/5$

e agora o erro é:

$$Er_{\mu}(\varphi, \psi) = \frac{12}{5},$$

o qual é quase a metade do erro anterior.

Estas tabelas mostram que a hipótese ϕ está "mais perto" de ψ que a hipótese φ, mas o sistema gerado por ϕ é pelo menos duas vezes pior que o sistema gerado por φ, quando as funções são aplicadas iterativamente. É interessante notar que

$$Er_{\mu=}(\phi, \psi, 2) = Er_{\mu}(\varphi, \psi, 2) = \frac{6}{5},$$

e assim, até a segunda iteração a função erro não diferencia ϕ de φ. ■

Seja $S(mk, \psi)$ o conjunto de amostras de treinamento de m condições iniciais para um FDS dado com função de transição ψ, onde cada amostra consiste de um estado inicial e

as prósimas k iterações de ψ. Ou seja, um elemento $s \in S(mk, \psi)$ é da forma:

$$s = ((x_1, \psi(x_1), \ldots, \psi^{k-1}(x_1)), \ldots, (x_m, \psi(x_m), \ldots, \psi^{k-1}(x_m))).$$

Assím, temos uma bijeção natural $\theta\colon X^m \longrightarrow S(mk, \psi)$ dada por $(x_1, \ldots, x_m) \mapsto s$. Simplesmente é medido o conjunto de amostras $s \in S(mk, \psi)$ com a propriedade P por

$$\mu^m(\theta^{-1}\{s \in S(mk, \psi) \ : \ s \ \text{ tem a propriedade } \ P\}).$$

Um *algoritmo de identificação* L de $S(mk, \psi)$ em H é chamado *Provavelmente Aproximadamente Correto (PAC)* se, para um $k \geq 2$ fixo, existe $m \geq m_0(\epsilon, \delta)$ tal que

$$\mu^m(\theta^{-1}\{s \in S(mk, \psi) : \frac{1}{k} Er_\mu(L(s), \psi, k) < \epsilon \}) > 1 - \delta,$$

para quaisquer números reais $\epsilon, \delta \in (0, 1)$, distribuição μ sobre X, e FDS alvo $S(\psi)$, com $\psi \in H$. Note que o termo $\frac{1}{k}$ normaliza o erro, isto é,

$$0 \quad \leq \quad \frac{1}{k} Er_\mu(L(s), \psi, k) \quad \leq \quad 1$$

Um algoritmo de identificação L para o espaço de hipótese H é consistente se, dada qualquer amostra de treinamento s para um FDS alvo $S(\psi)$, com $\psi \in H$, a hipótese de saída $S(\phi)$ é tal que $\phi = L(s) \in H$ concorda com ψ em s.

Se H é um espaço de hipótese finito e L é um algoritmo de identificação consistente, então L é PAC e [13]

$$m_0(\epsilon, \delta) = \frac{(k-1)}{\epsilon} \ln(\frac{|H|}{\delta}).$$

Veja que para $k = 2$, a última fórmula de complexidade de identificação se reduz à expressão clássica de complexidade do aprendizado PAC. Esta fórmula está longe de constituir um limite refinado, por ter sido derivada em condições bastante gerais, mas mostra claramente que o resultado da identificação piora quando o tamanho do espaço de hipóteses aumenta. Assim, um jeito de simplificar o problema de identificação é escolher o menor espaço de hipóteses que contém o FDS alvo ou, pelo menos, um FDS muito similar ao alvo. Pode-se diminuir o tamanho do espaço de hipóteses mediante a aplicação de restrições, como por exemplo usando a técnica de envelopes dinâmicos.

4.2.1 Identificação de um Sistema Booleano

Consideramos aqui a identificação do sistema descrito na Seção 3.2. Como cada um dos componentes de $S(\Phi)$ é uma função Booleana, a identificação do sistema pode ser vista como um problema de identificar $n = 5$ funções booleanas, cada uma delas correspondendo a cada uma das componentes do vetor de estados.

O processo de treinamento consta dos seguintes passos:

- Escolher aleatoriamente exemplos extraídos dos resultados da simulação. Uma amostra de treinamento é uma seqüência de estados de comprimento 7 de uma trajetória, onde os primeiros 6 estados correspondem a uma entrada da função de transição e o último à sua respectiva saída.

- A partir destes exemplos, gerar exemplos de entrada-saída para cada uma das componentes do vetor de estados. Por exemplo, seja $(x[t-6], x[t-5], x[t-4], \ldots, x[t-1], x[t], y[t])$ um destes exemplos. Se a arquitetura é desconhecida, então o exemplo correspondente para a componente j é $(x[t-6], x[t-5], x[t-4], \ldots, x[t-1], x[t], y_j[t])$

($y_j[t]$ corresponde à componente j da saída y no instante t). Se a arquitetura é conhecida, então a entrada do exemplo correspondente é formada somente pelos estados de interesse. Por exemplo, para a componente x_1, os únicos elementos que são de interesse são: $x_1[t]$, $x_3[t-4]$, $x_3[t-3], x_3[t-2], x_3[t-1], x_3[t]$, $x_4[t-4], x_4[t-3], x_4[t-2], x_4[t-1], x_4[t]$ (ou seja, seu valor atual e os últimos 5 valores das componentes x_3 e x_4, respectivamente).

- Para cada exemplo de entrada, decidir se deve ser atribuída a saída 0 ou 1 (para cada componente j). No caso particular considerado aqui, a decisão é óbvia porque o valor de saída associado a uma entrada dada é sempre 0 ou sempre 1. Se não, a escolha da saída mais freqüente minimizaria o erro quadrático médio (MAE: "mean square error").

- O conjunto de exemplos de entrada-saída obtidos desta forma (para cada componente j) é uma função Booleana (parcialmente definida). Se deve usar algum algoritmo de aprendizado para minimizar sua representação e fazer a generalização (ou seja, atribuir valores de saída às entradas não observadas). A minimização de funções Booleanas, considerando as entradas não observadas como "don't cares", pode ser usada para este fim [31]. A forma mínima da função Booleana define a base de um operador.

Duas medidas de erro são consideradas para medir quão bem o sistema identificado realmente se parece ao original: *erro do sistema* e *erro de transição*. O erro definido em 4.2 é um erro do sistema para uma trajetória de comprimento k estados.

O erro do sistema calcula o erro médio absoluto, entre os estados dos sistemas original e identificado, considerando a trajetória total, neste caso, se o o sistema identificado erra

no começo de uma trajetória, então o erro se propagará ao resto dela.

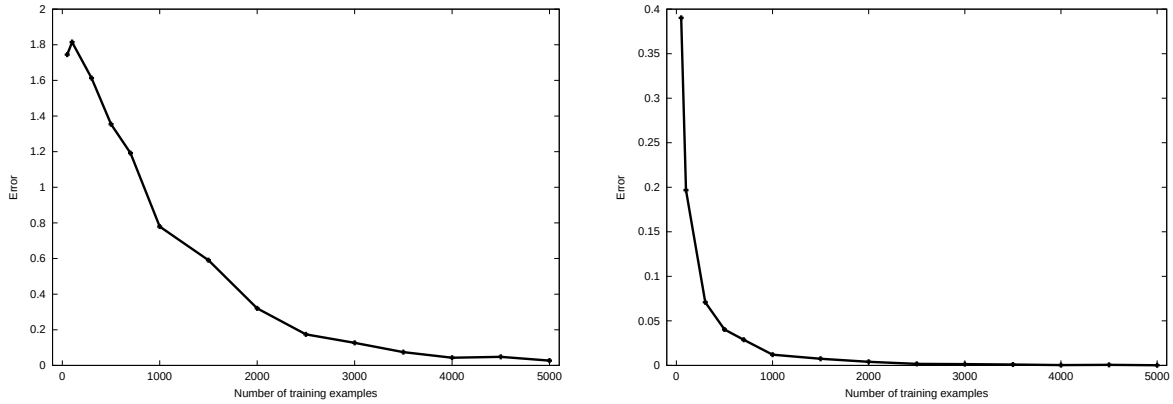
O erro de transição calcula o erro médio absoluto, também entre os estados dos sistemas original e identificado, mas considerando a transição passo a passo, ou seja, toma cada instante de tempo como uma condição inicial e avalia se o sistema identificado é capaz de predizer corretamente o próximo estado do sistema.

Por simplicidade, consideramos que somente dez condições iniciais são possíveis e que elas são igualmente prováveis. Os dados de treinamento são escolhidos aleatoriamente das trajetórias geradas a partir daquelas condições iniciais. Os erros são avaliados em todas as dez trajetórias.

Foi efetuada uma simulação de comprimento 100 (passos ou estados sucessivos) para 10 condições iniciais geradas aleatoriamente. Os dados de treinamento foram aleatoriamente extraídos das trajetórias simuladas. O sistema identificado foi simulado a partir das mesmas condições iniciais, efetuando-se as medidas de erro entre as trajetórias do sistema inicial (ideal) e o sistema identificado. O processo foi repetido 10 vezes fazendo-se a média das medidas de erro.

O processo de identificação foi aplicado para uma quantidade crescente de dados de treinamento, extraídos aleatoriamente das dez trajetórias. A Figura 4.1 mostra como variam os erros (do sistema e de transição) quando varia o tamanho da amostra de treinamento.

Analisando a curva de erro da Figura 4.1(b), vemos que com um número relativamente pequeno de dados de treinamento, quase todas as transições de um estado para o seguinte são aprendidas corretamente. No entanto, a curva de erro da Figura 4.1(a) indica que erros em algum ponto da trajetória (provavelmente no começo) se propagaram ao resto



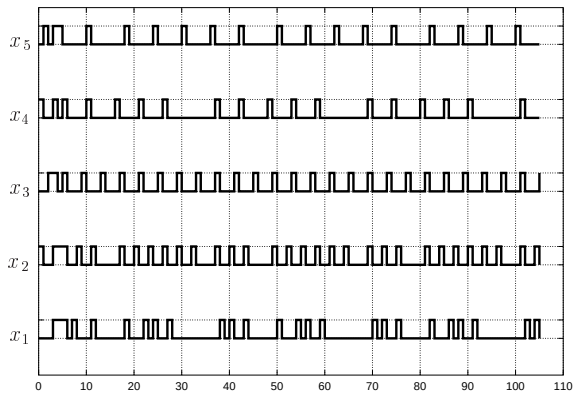
(a) Erro do Sistema

(b) Erro de Transição

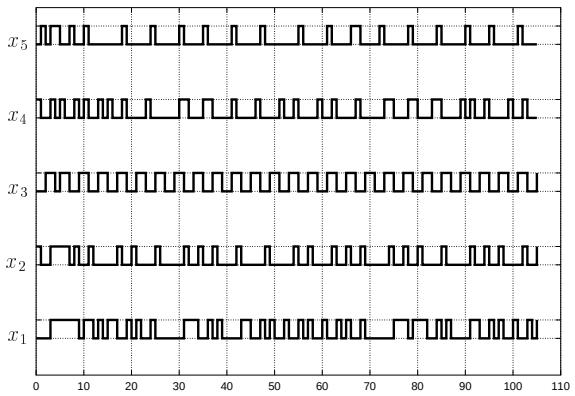
Figura 4.1: Medidas de erro para amostras de treinamento crescentes.

dela.

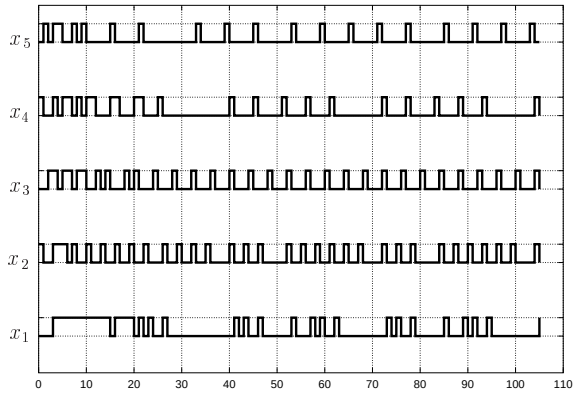
A Figura 4.2 mostra a simulação do sistema original e outras três simulações partindo das mesmas condições iniciais, cada uma delas geradas, respectivamente, por três sistemas identificados usando diferente quantidade de dados de treinamento. Vemos que a medida que o número de dados de treinamento aumenta, as trajetórias resultam mais parecidas à trajetória correta.



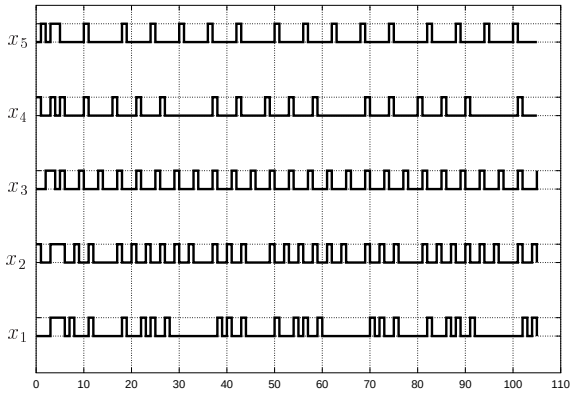
(a) Original



(b) 100 exemplos de treinamento



(c) 500 exemplos de treinamento



(d) 1500 exemplos de treinamento

Figura 4.2: Simulações dos Sistemas Original e Identificados.

Capítulo 5

Identificação com Restrição de Envelope

5.1 Restrição de Envelope Dinâmico

Seja L um reticulado completo, com uma relação de ordem parcial $\leq$, ínfimo $\wedge$ e supremo $\vee$. Sejam α e β dois operadores definidos em L. O par de operadores de reticulados (α, β) constitui um *envelope dinâmico* se e somente se, para todo $i \geq 1$ e $x \in L$, $\alpha^i(x) \leq \beta^i(x)$ [30].

Um operador de reticulados ϕ é *crescente* se, e somente se,

$$\forall x, y \in X, x \leq y \Rightarrow \phi(x) \leq \phi(y)$$

A seguinte proposição fornece duas condições suficientes para construir envelopes dinâmicos.

Proposição. Sejam α e β dois operadores de reticulados. O par $(\alpha,\ \beta)$ constitui um envelope dinâmico se uma das seguintes condições é satisfeita [30]:

1. $\alpha \leq \beta$ e β é crescente;

2. $\alpha \leq \beta$ e α é crescente.

Dem.: *Provemos a condição 1. Como* $\alpha \leq \beta$, *temos que* $\alpha(x_0) \leq \beta(x_0)$ *e, dado que* β *é crescente,* $\beta(\alpha(x_0)) \leq \beta(\beta(x_0))$. *Novamente, dado que* $\alpha \leq \beta$, *temos que* $\alpha(\alpha(x_0)) \leq \beta(\alpha(x_0)) \leq \beta(\beta(x_0))$. *Assim,* $\alpha^2(x_0) \leq \beta^2(x_0)$. *Este é o passo inicial da prova por indução. O passo da indução (i.e.,* $\alpha^{i-1}(x_0) \leq \beta^{i-1}(x_0)$ *implica que* $\alpha^i(x_0) \leq \beta^i(x_0)$*) é provado usando os mesmos argumentos. A condição 2 pode ser provada com argumentos similares.* ■

Sejam α e β elementos do espaço de hipóteses H tais que (α, β) é um envelope dinâmico. O par (α, β) define um subconjunto H' do espaço de hipóteses H tal que $\phi \in H' \Leftrightarrow \alpha^i \leq \phi^i \leq \beta^i$, para todo $i \geq 1$.

A identificação do sistema dinâmico finito alvo $S(\psi)$, com $\psi \in H$, restrito a H', pode ser implementada como um processo de dois passos:

1. estimar a hipótese $\phi \in H$ para ψ,

2. construir a hipótese θ em H', isto é,

   para todo $i \geq 1$,

$$\theta^i = (\phi^i \wedge \beta^i) \vee \alpha^i$$

5.2 Resultados Experimentais

Sejam $K = [-100, ..., 0, ..., 100]$ a escala de valores e $X = K^2$ o espaço de estados. Os operadores ψ, α e β definidos de X^2 em X são as funções de transição do FDS alvo e das extremidades superior e inferior do envelope, respectivamente. A Figura 5.1 mostra

a simulação destes sistemas. As duas curvas de alta freqüência são ψ_1 e ψ_2, as componentes de ψ. As curvas superior e inferior são, respectivamente, $\alpha_1 = \alpha_2$ e $\beta_1 = \beta_2$, as componentes das extremidades inferior e superior da envelope dinâmico.

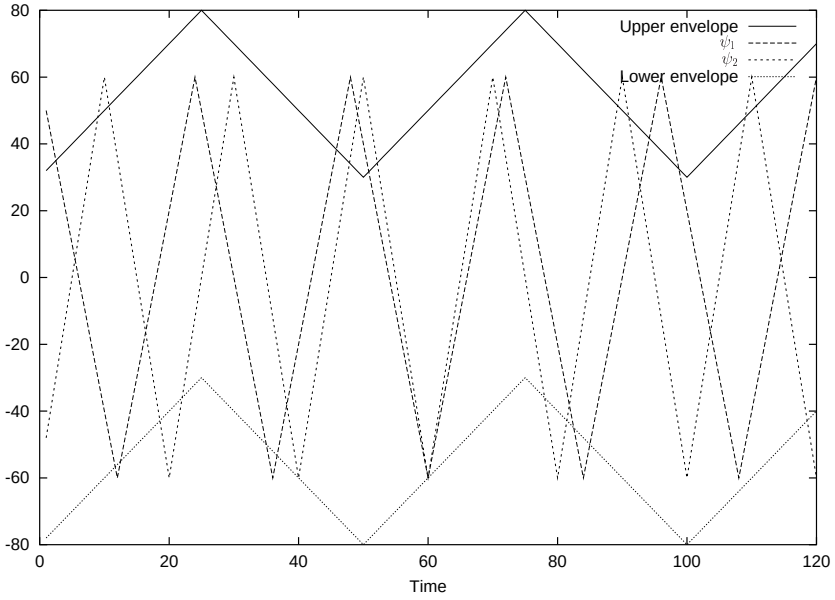


Figura 5.1: Simulação do Sistema 1.

Escolhimos aleatoriamente 15 condições iniciais e simulamos 120 iterações da função de transição ψ do FLDS alvo a partir de cada condição inicial. Depois disso, extraímos aleatoriamente 50 exemplos do tipo $(x_i, \psi(x_i))$ e identificamos ϕ usando o algoritmo ISI de níveis de cinza (GISI: "Gray-scale Incremental Splitting of Intervals") [25], que gera uma coleção minimal de intervalos maximais que representam a base das componentes de ϕ. Depois, foram simuladas as dinâmicas ϕ^i e $(\phi^i \wedge \beta^i) \vee \alpha^i$.

Finalmente, considerando a função de perda do *erro médio quadrático* (MSE: "Mean Square Error"), o erro entre ψ^i e ϕ^i, e entre ψ^i e $(\phi^i \wedge \beta^i) \vee \alpha^i$ foi estimado, sobre as simulações de todas as 15 condições iniciais.

A figura 5.2 mostra a curva de erro para as duas hipóteses, ϕ^i e $(\phi^i \wedge \beta^i) \vee \alpha^i$,

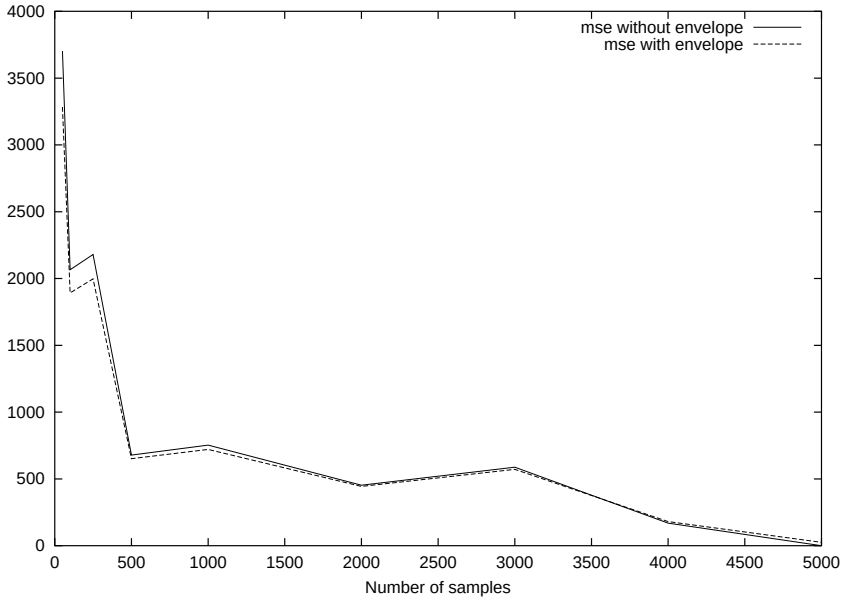


Figura 5.2: Curvas de erro para o Sistema 1.

estimadas com amostras de tamanho crescente. Note que para amostras pequenas o erro de $(\phi^i \wedge \beta^i) \vee \alpha^i$ é menor do que o erro de ϕ^i. No entanto, para amostras grandes, a situação é invertida.

Este experimento foi repetido para o mesmo FLDS alvo ψ, mas com um envelope dinâmico mais severa, ou seja, um limite inferior maior e um limite superior menor. As Figuras 5.3 e 5.4 mostram, respectivamente, a simulação destes sistemas e as correspondentes curvas de erro. O mesmo tipo de fenômeno observado no primeiro experimento acontece de novo, mas esta vez as diferenças de erro entre ϕ^i e $(\phi^i \wedge \beta^i) \vee \alpha^i$ são mais significativas.

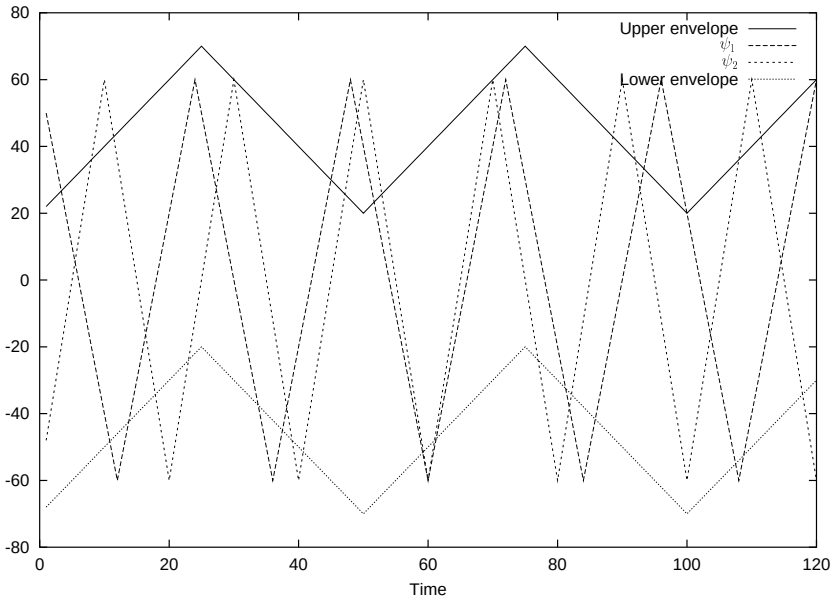


Figura 5.3: Simulação do Sistema 2.

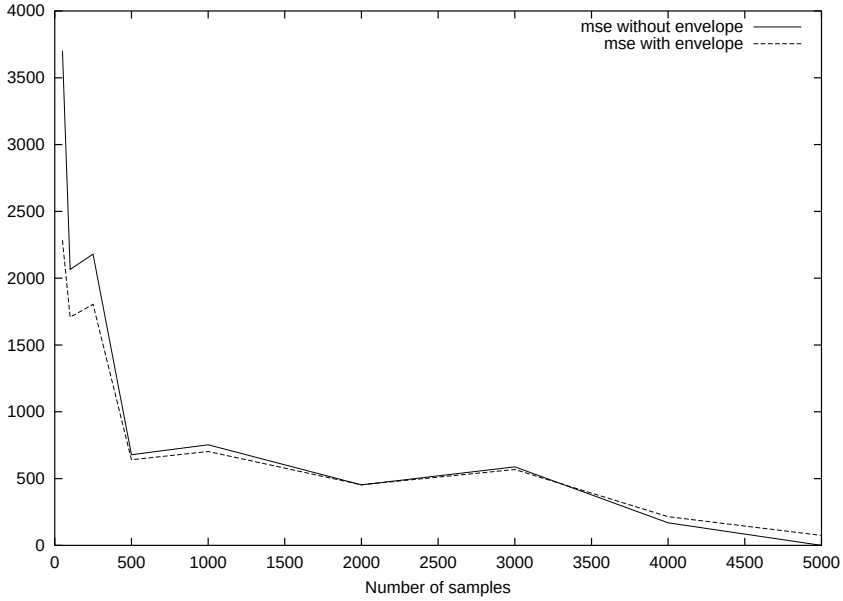


Figura 5.4: Curvas de Erro para o Sistema 2.

Capítulo 6

Conclusão

Foram estudadas e desenvolvidas técnicas de modelagem, simulação, aprendizado computacional e identificação de sistemas dinâmicos finitos de reticulados.

Sob esta estrutura algébrica foi modelado e simulado um ciclo celular hipotético, fazendo que o seu comportamento represente hipóteses acerca das características da dinâmica de um sistema biológico real. Ele mostrou aspectos interessantes dos sistemas dinâmicos como robustez, flexibilidade, adaptabilidade, etc.. O sistema foi modelado com valores binários e em níveis de cinza. A modelagem foi feita interativamente com o simulador.

Foi estudada a identificação de FLDSs livres de entradas a partir de amostras de suas dinâmicas, com e sem a aplicação da *restrição de envelope dinâmico*. Os resultados experimentais mostraram que a restrição de envelope pode ser *benéfica para amostras pequenas*.

Na identificação, a *estrutura de reticulados* resulta importante para: *i* - representar as componentes da função de transição por uma coleção minimal de intervalos maximais, que é gerada pelo algoritmo GISI; *ii* - construir um envelope dinâmico; *iii* - projetar o

sistema identificado no espaço restrito mediante as operações de ínfimo e supremo.

O uso da estrutura algébrica de FLDS reduz o problema de identificação do sistema a uma coleção de problemas de projeto de operadores de reticulados. Assim, o problema de identificação do sistema dinâmico é reduzido a problemas clássicos de reconhecimento de padrões.

A técnica de identificação sob restrição de envelope dinâmico pode ser útil na modelagem híbrida de FLDSs. O pesquisador usa o seu conhecimento acerca do FLDS alvo para idealizar heuristicamente uma hipótese, depois flexibiliza o seu modelo para construir um envelope. Os perfis de expressão devem ser usados para estimar as funções de transição, que serão corrigidas pelo envelope dinâmico.

Desta forma, a identificação a partir de amostras de um FLDS alvo, sob esta restrição de envelope, permite a integração de conhecimento prévio com a informação presente nos dados.

A modelagem de um sistema dinâmico que descreve o comportamento de um sistema biológico real é um problema muito complexo que requer o uso de toda a informação disponível: conhecimento biológico prévio e dados experimentais. Usualmente o número de genes envolvidos é muito grande, o que dificulta ou impede a modelagem.

A simulação do sistema identificado (restrito ao envelope dinâmico) pode sugerir novas interpretações biológicas e novos experimentos a serem realizados. O uso do novo conhecimento e dos dados, mediante a interação com o simulador deve levar a melhorias sucessivas do modelo e do conhecimento que se tem do sistema alvo.

Um interessante trabalho a ser realizado no futuro como continuação do presente é a identificação da dinâmica do ciclo celular modelado a partir de amostras de seus estados.

A técnica de aprendizado desenvolvida, referente a geração automática de árvores de classificação por multirresolução, pode ser utilizada no futuro na identificação de sistemas dinâmicos finitos, onde seja conveniente o uso de uma técnica de multiclassificação especialmente adaptada a um problema específico de classificação.

Novas técnicas de *identificação com restrições* serão exploradas e estudar-se-á a *identificação da arquitetura* das redes de expressão gênica consideradas como um sistema dinâmico finito.

A identificação da arquitetura é importante para inferir as dependências funcionais entre as variáveis de estado (níveis de expressão dos genes, em nosso caso). Dada a complexidade destas redes, é necessário também determinar quais destas dependências são mais relevantes, ou seja, quais tem maior influência no comportamento da rede, a fim de simplificar o modelo e ter maior precisão na identificação. Ao desprezar as dependências mais fracas, espera-se minimizar a perda de informação sobre o comportamento dinâmico do sistema.

Também será estudada no futuro a identificação de sistemas dinâmicos finitos estocásticos. Eles constituem uma generalização probabilística dos FDSs, que podem representar a incerteza existente nas trajetórias de estados e nos parâmetros das funções que determinam as trajetórias de estado. Os sistemas estocásticos constituem uma abordagem mais flexível para modelar sistemas biológicos.

Apêndice A

Modelo Binário do Ciclo Celular

A.1 Arquitetura

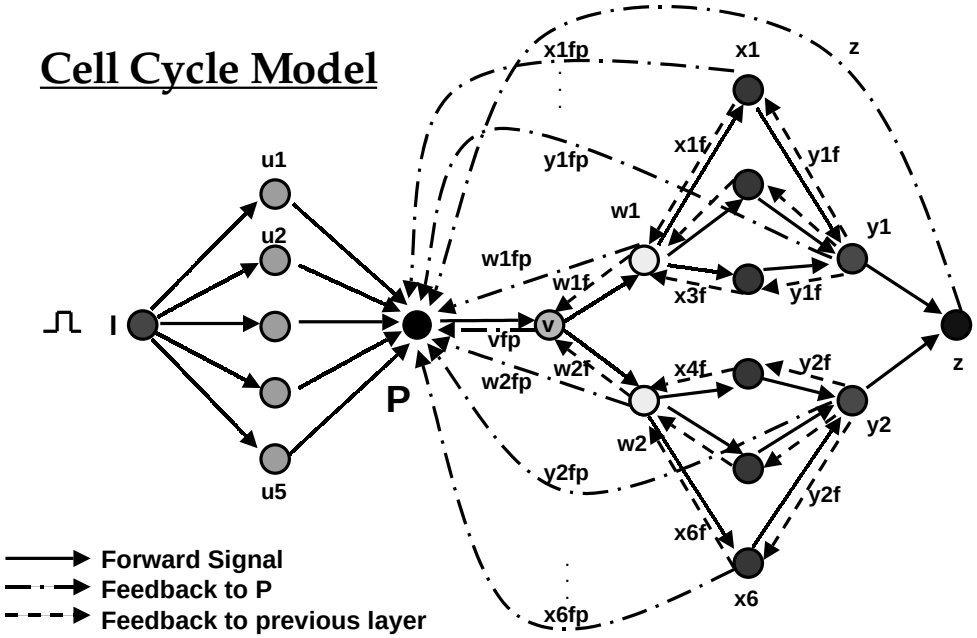


Figura A.1: Arquitetura do Modelo de Ciclo Celular.

A.2 Definição das Funções de Transição e Realimentação

A.2.1 Genes $\mathbf{u_j}$, $j = 1, 2, \ldots, 5$

$$\mathbf{u_j = 1} \quad \Leftrightarrow \quad \text{pelo menos 3 dos } I[t-i] = 1 \quad \text{e} \quad N_j = 1, \quad i = 1, 2, \ldots, 5$$

onde $N_j = 1$ significa que existe suficiente substância
e $N_j = 1 \quad \Leftrightarrow \quad n_{j1} \geq t_{j1} \quad \text{e} \quad n_{j2} \geq t_{j2} \quad \text{e} \quad n_{j3} \geq t_{j3}$
$t_{j1}t_{j2}t_{j3}$: limiar de substância para u_j
$n_{j1}n_{j2}n_{j3}$: osciladores de período aleatoriamente variável
I : sinal de excitação externo

A.2.2 Gene P

$$\mathbf{P = 1} \quad \text{se, e somente se,}$$

\# caso 1: ***sem sinal*** *através do ciclo celular*

$$\text{pelo menos 1 dos } FP[t-i] = 1, \qquad (i = 1, \ldots, 5)$$

$$\begin{aligned}
&\text{e} \quad v_{fp}[t-j] = 0, && (\forall \quad j = 1, 2) \\
&\text{e} \quad w_{kfp}[t-l] = 0, && (\forall \quad l = 1, 2; \quad k = 1, 2) \\
&\text{e} \quad x_{mfp}[t-n] = 0, && (\forall \quad n = 1, 2; \quad m = 1, \ldots, 6) \\
&\text{e} \quad y_{ofp}[t-p] = 0, && (\forall \quad p = 1, 2; \quad o = 1, 2) \\
&\text{e} \quad z[t-1] = 0 \\
&\text{e} \quad P[t-1] = 0
\end{aligned}$$

ou

caso 2: *sinal na* ***camada 'v'*** *do ciclo celular*

$$\text{pelo menos 3 dos } FP[t-i]=1, \qquad (i=1,\ldots,5)$$

$$\begin{aligned}
&\text{e} \quad v_{fp}[t-j]=1, && (\forall \quad j=1,2)\\
&\text{e} \quad w_{kfp}[t-l]=0, && (\forall \quad l=1,2; \quad k=1,2)\\
&\text{e} \quad x_{mfp}[t-n]=0, && (\forall \quad n=1,2; \quad m=1,\ldots,6)\\
&\text{e} \quad y_{ofp}[t-p]=0, && (\forall \quad p=1,2; \quad o=1,2)\\
&\text{e} \quad z[t-1]=0,\\
&\text{e} \quad P[t-1]=0
\end{aligned}$$

ou

caso 3: *sinal na* ***camada 'w'*** *do ciclo celular*

$$\text{pelo menos 4 dos } FP[t-i]=1 \qquad (i=1,\ldots,5)$$

$$\begin{aligned}
&\text{e} \quad v_{fp}[t-j]=1 && (\forall \quad j=1,2)\\
&\text{e} \quad w_{kfp}[t-l]=1 && (\forall \quad l=1,2; \quad k=1,2)\\
&\text{e} \quad x_{mfp}[t-n]=0 && (\forall \quad n=1,2; \quad m=1,\ldots,6)\\
&\text{e} \quad y_{ofp}[t-p]=0 && (\forall \quad p=1,2; \quad o=1,2)\\
&\text{e} \quad z[t-1]=0\\
&\text{e} \quad P[t-1]=0
\end{aligned}$$

ou

caso 4: *sinal* ***camada 'x'*** *do ciclo celular*

$$\text{pelo menos 5 dos } FP[t-i]=1 \qquad (i=1,\ldots,5)$$

$$\begin{array}{lll}
\text{e} & v_{fp}[t-j]=0 & (\forall \quad j=1,2) \\
\text{e} & w_{kfp}[t-l]=1 & (\forall \quad l=1,2; \quad k=1,2) \\
\text{e} & x_{mfp}[t-n]=1 & (\forall \quad n=1,2; \quad m=1,\ldots,6) \\
\text{e} & y_{ofp}[t-p]=0 & (\forall \quad p=1,2; \quad o=1,2) \\
\text{e} & z[t-1]=0 &
\end{array}$$

$$\text{e} \quad P[t-1]=0$$

ou

caso 5: *sinal* ***camada 'y'*** *do ciclo celular*

$$\text{pelo menos 4 dos } FP[t-i]=1 \qquad (i=1,\ldots,5)$$

$$\begin{array}{lll}
\text{e} & v_{fp}[t-j]=0 & (\forall \quad j=1,2) \\
\text{e} & w_{kfp}[t-l]=0 & (\forall \quad l=1,2; \quad k=1,2) \\
\text{e} & x_{mfp}[t-n]=1 & (\forall \quad n=1,2; \quad m=1,\ldots,6) \\
\text{e} & y_{ofp}[t-p]=1 & (\forall \quad p=1,2; \quad o=1,2) \\
\text{e} & z[t-1]=0 &
\end{array}$$

$$\text{e} \quad P[t-1]=0$$

ou

caso 6: *sinal* ***camada 'z'*** *do ciclo celular*

$$\text{pelo menos 3 dos } FP[t-i]=1 \qquad (i=1,\ldots,5)$$

$$\begin{array}{lll} \text{e} & v_{fp}[t-j]=0 & (\forall \quad j=1,2) \\ \text{e} & w_{kfp}[t-l]=0 & (\forall \quad l=1,2; \quad k=1,2) \\ \text{e} & x_{mfp}[t-n]=0 & (\forall \quad n=1,2; \quad m=1,\ldots,6) \\ \text{e} & y_{ofp}[t-p]=0 & (\forall \quad p=1,2; \quad o=1,2) \\ \text{e} & z[t-1]=1 & \end{array}$$

$$\text{e} \quad P[t-1]=0$$

onde:

$$\begin{array}{lll} FP=1 & \Leftrightarrow & \text{pelo menos 3 dos } Fu_q=1, \quad q=1,2,\ldots,5 \\ \text{e} \quad Fu_q=1 & \Leftrightarrow & \text{pelo menos 3 dos } u_q[t-r]=1, \quad r=2,3,\ldots,6 \end{array}$$

A.2.3 Gene v

$$\mathbf{v=1} \quad \Leftrightarrow$$

$$\begin{array}{ll} ((\text{pelo menos 1 dos } P[t-i]=1 & i=1,2,\ldots,5 \\ \quad \text{e} \quad w_{jf}[t-k]=0) & \forall \quad j=1,2; \quad k=1,2 \\ \quad \text{ou} & \\ \text{pelo menos 2 dos } P[t-l]=1) & l=1,\ldots,5 \end{array}$$

$$\text{e} \quad v[t-1]=0$$

$$\mathbf{v_{fp}} = \mathbf{1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } P[t-i]=1 & i=1,2,\ldots,5 \\
\text{e} \quad w_{jf}[t-k]=0) & \forall \quad j=1,2; \quad k=1,2 \\
\text{ou} & \\
\text{pelo menos 2 dos } P[t-l]=1) & l=1,\ldots,5
\end{array}$$

$$\text{e} \quad v[t-m]=0 \qquad \text{para } m=3,4$$

A.2.4 Genes w

Gene $\mathbf{w_1}$

$$\mathbf{w_1 = 1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } v[t-i] = 1 & i = 1, 2, \ldots, 5 \\
\text{e} \quad x_{kf}[t-l] = 0) & \forall \quad k = 1, 2, 3; \quad l = 1, 2 \\
\text{ou} & \\
\text{pelo menos 3 dos } v[t-m] = 1) & m = 1, \ldots, 5
\end{array}$$

$$\text{e} \quad w_1[t-1] = 0$$

$$\mathbf{w_{1f} = 1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } v[t-i] = 1 & i = 1, 2, \ldots, 5 \\
\text{e} \quad x_{kf}[t-l] = 0) & \forall \quad k = 1, 2, 3; \quad l = 1, 2 \\
\text{ou} & \\
\text{pelo menos 3 dos } v[t-m] = 1) & m = 1, \ldots, 5
\end{array}$$

$$\begin{array}{l}
\text{ou} \quad w_1[t-1] = 1 \\
\text{ou} \quad w_1[t-2] = 1
\end{array}$$

$$\mathbf{w_{1fp} = 1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } v[t-i] = 1 & i = 1, 2, \ldots, 5 \\
\text{e} \quad x_{kf}[t-l] = 0) & \forall \quad k = 1, 2, 3; \quad l = 1, 2 \\
\text{ou} & \\
\text{pelo menos 3 dos } v[t-m] = 1) & m = 1, \ldots, 5 \\
 & \\
\text{e} \quad w_1[t-n] = 0 & \text{para } n = 2, 3
\end{array}$$

Gene $\mathbf{w_2}$

$$\mathbf{w_2 = 1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } v[t-i] = 1 & i = 1,2,\ldots,5 \\
\text{e} \quad x_{kf}[t-l] = 0) & \forall \quad k = 4,5,6; \quad l = 1,2 \\
\text{ou} & \\
\text{pelo menos 3 dos } v[t-m] = 1) & m = 1,\ldots,5
\end{array}$$

$$\text{e} \quad w_2[t-1] = 0$$

$$\mathbf{w_{2f} = 1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } v[t-i] = 1 & i = 1,2,\ldots,5 \\
\text{e} \quad x_{kf}[t-l] = 0) & \forall \quad k = 4,5,6; \quad l = 1,2 \\
\text{ou} & \\
\text{pelo menos 3 dos } v[t-m] = 1) & m = 1,\ldots,5
\end{array}$$

$$\begin{array}{l}
\text{ou} \quad w_2[t-1] = 1 \\
\text{ou} \quad w_2[t-2] = 1
\end{array}$$

$$\mathbf{w_{2fp} = 1} \quad \Leftrightarrow$$

$$\begin{array}{rr}
((\text{pelo menos 1 dos } v[t-i] = 1 & i = 1,2,\ldots,5 \\
\text{e} \quad x_{kf}[t-l] = 0) & \forall \quad k = 4,5,6; \quad l = 1,2 \\
\text{ou} & \\
\text{pelo menos 3 dos } v[t-m] = 1) & m = 1,\ldots,5 \\
 & \\
\text{e} \quad w_2[t-n] = 0 & \text{para } n = 2,3
\end{array}$$

A.2.5 Genes x

Gene x_i, (para cada $i = 1, 2, 3$ –fixo–)

$$\mathbf{x_i = 1} \quad \Leftrightarrow$$

$$\begin{array}{cc} ((\text{pelo menos 1 dos } w_1[t-j] = 1 & j = 1, 2, \ldots, 5 \\ \text{e} \quad y_{1f}[t-k] = 0) & \forall \quad k = 1, 2 \\ \text{ou} & \\ \text{pelo menos 3 dos } w_1[t-l] = 1) & l = 1, \ldots, 5 \end{array}$$

$$\text{e} \quad x_i[t-1] = 0$$

$$\mathbf{x_{if} = 1} \quad \Leftrightarrow$$

$$\begin{array}{cc} ((\text{pelo menos 1 dos } w_1[t-j] = 1 & j = 1, 2, \ldots, 5 \\ \text{e} \quad y_{1f}[t-k] = 0) & \forall \quad k = 1, 2 \\ \text{ou} & \\ \text{pelo menos 3 dos } w_1[t-l] = 1) & l = 1, \ldots, 5 \end{array}$$

$$\begin{array}{l} \text{ou} \quad x_i[t-1] = 1 \\ \text{ou} \quad x_i[t-2] = 1 \end{array}$$

$$\mathbf{x_{ifp} = 1} \quad \Leftrightarrow$$

$$\begin{array}{cc} ((\text{pelo menos 1 dos } w_1[t-j] = 1 & j = 1, 2, \ldots, 5 \\ \text{e} \quad y_{1f}[t-k] = 0) & \forall \quad k = 1, 2 \\ \text{ou} & \\ \text{pelo menos 3 dos } w_1[t-l] = 1) & l = 1, \ldots, 5 \end{array}$$

$$\text{e} \quad x_i[t-m] = 0 \qquad \text{para } m = 2, 3$$

Gene $\mathbf{x_i}$, (para cada $i = 4, 5, 6$ –fixo–)

$$\mathbf{x_i = 1} \quad \Leftrightarrow$$

$$\begin{array}{ll} ((\text{pelo menos 1 dos } w_2[t-j] = 1 & j = 1, 2, \ldots, 5 \\ \quad \text{e} \quad y_{2f}[t-k] = 0) & \forall \quad k = 1, 2 \\ \quad \text{ou} & \\ \text{pelo menos 3 dos } w_2[t-l] = 1) & l = 1, \ldots, 5 \end{array}$$

$$\text{e} \quad x_i[t-1] = 0$$

$$\mathbf{x_{if} = 1} \quad \Leftrightarrow$$

$$\begin{array}{ll} ((\text{pelo menos 1 dos } w_2[t-j] = 1 & j = 1, 2, \ldots, 5 \\ \quad \text{e} \quad y_{2f}[t-k] = 0) & \forall \quad k = 1, 2 \\ \quad \text{ou} & \\ \text{pelo menos 3 dos } w_2[t-l] = 1) & l = 1, \ldots, 5 \end{array}$$

$$\begin{array}{l} \text{ou} \quad x_i[t-1] = 1 \\ \text{ou} \quad x_i[t-2] = 1 \end{array}$$

$$\mathbf{x_{ifp} = 1} \quad \Leftrightarrow$$

$$\begin{array}{ll} ((\text{pelo menos 1 dos } w_2[t-j] = 1 & j = 1, 2, \ldots, 5 \\ \quad \text{e} \quad y_{2f}[t-k] = 0) & \forall \quad k = 1, 2 \\ \quad \text{ou} & \\ \text{pelo menos 3 dos } w_2[t-l] = 1) & l = 1, \ldots, 5 \end{array}$$

$$\text{e} \quad x_i[t-m] = 0 \qquad \text{para } m = 2, 3$$

A.2.6 Genes y

Gene y_1

$$\mathbf{y_1 = 1} \quad \Leftrightarrow$$

$$x_1[t-1] = 1 \quad \text{e} \quad x_2[t-1] = 1 \quad \text{e} \quad x_3[t-1] = 1$$
$$\text{e} \quad y_1[t-1] = 0$$

$$\mathbf{y_{1f} = 1} \quad \Leftrightarrow$$

$$(x_1[t-1] = 1 \quad \text{e} \quad x_2[t-1] = 1 \quad \text{e} \quad x_3[t-1] = 1$$
$$\text{e} \quad y_1[t-1] = 0)$$
$$\text{ou} \quad y_1[t-1] = 1$$
$$\text{ou} \quad y_1[t-2] = 1$$

$$\mathbf{y_{1fp} = 1} \quad \Leftrightarrow$$

$$(x_1[t-1] = 1 \quad \text{e} \quad x_2[t-1] = 1 \quad \text{e} \quad x_3[t-1] = 1$$
$$\text{e} \quad y_1[t-1] = 0)$$
$$\text{e} \quad y_1[t-2] = 0$$

Gene $\mathbf{y_2}$

$$\mathbf{y_2 = 1} \quad \Leftrightarrow$$

$$x_4[t-1] = 1 \quad \text{e} \quad x_5[t-1] = 1 \quad \text{e} \quad x_6[t-1] = 1$$
$$\text{e} \quad y_2[t-1] = 0$$

$$\mathbf{y_{2f} = 1} \quad \Leftrightarrow$$

$$(x_4[t-1] = 1 \quad \text{e} \quad x_5[t-1] = 1 \quad \text{e} \quad x_6[t-1] = 1$$
$$\text{e} \quad y_2[t-1] = 0)$$
$$\text{ou} \quad y_2[t-1] = 1$$
$$\text{ou} \quad y_2[t-2] = 1$$

$$\mathbf{y_{2fp} = 1} \quad \Leftrightarrow$$

$$(x_4[t-1] = 1 \quad \text{e} \quad x_5[t-1] = 1 \quad \text{e} \quad x_6[t-1] = 1$$
$$\text{e} \quad y_2[t-1] = 0)$$
$$\text{e} \quad y_2[t-2] = 0$$

A.2.7 Gene z

$$\mathbf{z = 1} \quad \Leftrightarrow$$

$$y_1[t-1] = 1 \quad \text{e} \quad y_2[t-1] = 1$$
$$\text{e} \quad z[t-1] = 0$$

Apêndice B

Modelo em Níveis de Cinza do Ciclo Celular

B.1 Arquitetura

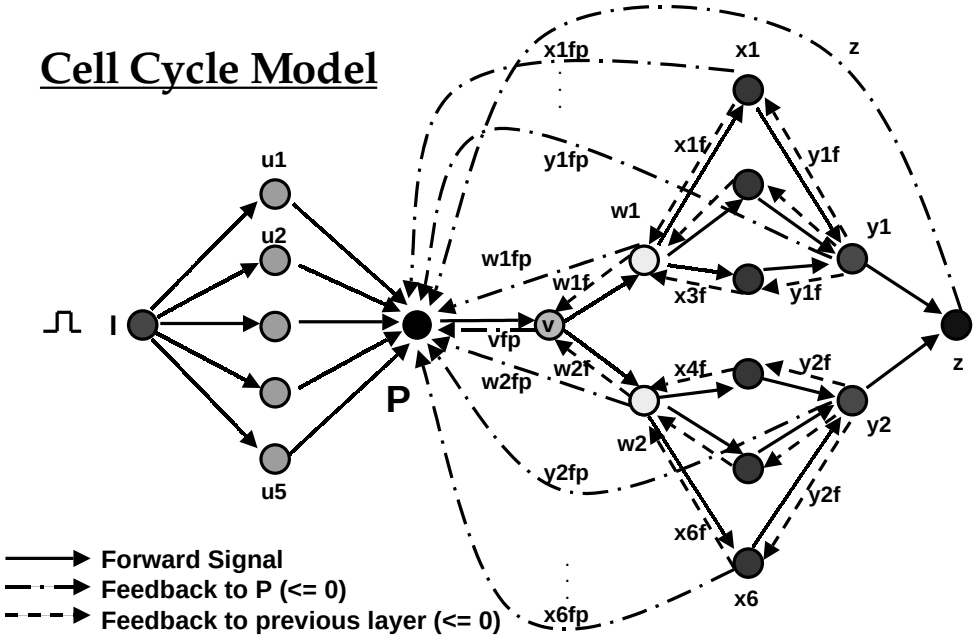


Figura B.1: Arquitetura do Modelo de Ciclo Celular.

B.2 Definição das Funções de Transição e Realimentação

B.2.1 Genes u

Gene $\mathbf{u_1}$

$$\mathbf{u_1} = \begin{cases} \mathbf{-2} & \text{se} \quad S = -10, \ldots, -4 \quad \text{e} \quad N_1 = 2 \\ \mathbf{-1} & \text{se} \quad S = -3, \ldots, 0 \quad \text{e} \quad N_1 = 2 \\ \mathbf{0} & \text{se} \quad (S = 1, \ldots, 4 \quad \text{e} \quad N_1 = 2) \quad \text{ou} \quad N_1 = -2 \\ \mathbf{1} & \text{se} \quad S = 5, \ldots, 8 \quad \text{e} \quad N_1 = 2 \\ \mathbf{2} & \text{se} \quad S = 9, 10 \quad \text{e} \quad N_1 = 2 \end{cases}$$

onde

$$S = \sum_{i=1}^{5} I[i]$$

e

$$N_1 = \begin{cases} 2 & \text{se} \quad n_{11}[t-1] \geq t_{11} \quad \text{e} \quad n_{12}[t-1] \geq t_{12} \quad \text{e} \quad n_{13}[t-1] \geq t_{13} \\ -2 & \text{caso contrário} \end{cases}$$

$N_1 = 2$ significa que existe *suficiente substância* para que o gene u_1 se expresse.
t_{11}, t_{12}, t_{13} : limiar de substância para u_1.
n_{11}, n_{12}, n_{13} : osciladores de período aleatoriamente variável.

Gene $\mathbf{u_2}$

$$\mathbf{u_2} = \begin{cases} \mathbf{-1} & \text{se} \quad S = -10, \ldots, -5 \quad \text{e} \quad N_2 = 2 \\ \mathbf{0} & \text{se} \quad (S = -6, \ldots, 1 \quad \text{e} \quad N_2 = 2) \quad \text{ou} \quad N_2 = -2 \\ \mathbf{1} & \text{se} \quad S = 2, \ldots, 7 \quad \text{e} \quad N_2 = 2 \\ \mathbf{2} & \text{se} \quad S = 8, 9, 10 \quad \text{e} \quad N_2 = 2 \end{cases}$$

onde

$$S = \sum_{i=1}^{5} I[i]$$

e

$$N_2 = \begin{cases} 2 & \text{se} \quad n_{21}[t-1] \geq t_{21} \quad \text{e} \quad n_{22}[t-1] \geq t_{22} \quad \text{e} \quad n_{23}[t-1] \geq t_{23} \\ -2 & \text{caso contrário} \end{cases}$$

$N_2 = 2$ significa que existe *suficiente substância* para que o gene u_2 se expresse.
t_{21}, t_{22}, t_{23} : limiar de substância para u_2.
n_{21}, n_{22}, n_{23} : osciladores de período aleatoriamente variável.

Gene $\mathbf{u_3}$

$$\mathbf{u_3} = \begin{cases} \mathbf{-1} & \text{se} \quad S = -10 & \text{e} \quad N_3 = 2 \\ \mathbf{0} & \text{se} \quad (S = -9, \ldots, -2 & \text{e} \quad N_3 = 2) \quad \text{ou} \quad N_3 = -2 \\ \mathbf{1} & \text{se} \quad S = -1, \ldots, 6 & \text{e} \quad N_3 = 2 \\ \mathbf{2} & \text{se} \quad S = 7, \ldots, 10 & \text{e} \quad N_3 = 2 \end{cases}$$

onde

$$S = \sum_{i=1}^{5} I[i]$$

e

$$N_3 = \begin{cases} 2 & \text{se} \quad n_{31}[t-1] \geq t_{31} \quad \text{e} \quad n_{32}[t-1] \geq t_{32} \quad \text{e} \quad n_{33}[t-1] \geq t_{33} \\ -2 & \text{caso contrário} \end{cases}$$

$N_3 = 2$ significa que existe *suficiente substância* para que o gene u_3 se expresse.
t_{31}, t_{32}, t_{33} : limiar de substância para u_3.
n_{31}, n_{32}, n_{33} : osciladores de período aleatoriamente variável.

Gene $\mathbf{u_4}$

$$\mathbf{u_4} = \begin{cases} \mathbf{-2} & \text{se} \quad S = -10, \ldots, -4 & \text{e} \quad N_4 = 2 \\ \mathbf{-1} & \text{se} \quad S = -3, \ldots, 0 & \text{e} \quad N_4 = 2 \\ \mathbf{0} & \text{se} \quad (S = 1, \ldots, 4 & \text{e} \quad N_4 = 2) \quad \text{ou} \quad N_4 = -2 \\ \mathbf{1} & \text{se} \quad S = 5, \ldots, 8 & \text{e} \quad N_4 = 2 \\ \mathbf{2} & \text{se} \quad S = 9, 10 & \text{e} \quad N_4 = 2 \end{cases}$$

onde

$$S = \sum_{i=1}^{5} I[i]$$

e

$$N_4 = \begin{cases} 2 & \text{se} \quad n_{41}[t-1] \geq t_{41} \quad \text{e} \quad n_{42}[t-1] \geq t_{42} \quad \text{e} \quad n_{43}[t-1] \geq t_{43} \\ -2 & \text{caso contrário} \end{cases}$$

$N_4 = 2$ significa que existe *suficiente substância* para que o gene u_4 se expresse.
t_{41}, t_{42}, t_{43} : limiar de substância para u_4.
n_{41}, n_{42}, n_{43} : osciladores de período aleatoriamente variável.

Gene $\mathbf{u_5}$

$$\mathbf{u_5} = \begin{cases} \mathbf{-1} & \text{se} \quad S = -10, \ldots, -5 \quad \text{e} \quad N_5 = 2 \\ \mathbf{0} & \text{se} \quad (S = -6, \ldots, 1 \quad \text{e} \quad N_5 = 2) \quad \text{ou} \quad N_5 = -2 \\ \mathbf{1} & \text{se} \quad S = 2, \ldots, 7 \quad \text{e} \quad N_5 = 2 \\ \mathbf{2} & \text{se} \quad S = 8, 9, 10 \quad \text{e} \quad N_4 = 2 \end{cases}$$

onde

$$S = \sum_{i=1}^{5} I[i]$$

e

$$N_5 = \begin{cases} 2 & \text{se} \quad n_{51}[t-1] \geq t_{51} \quad \text{e} \quad n_{52}[t-1] \geq t_{52} \quad \text{e} \quad n_{53}[t-1] \geq t_{53} \\ -2 & \text{caso contrário} \end{cases}$$

$N_5 = 2$ significa que existe *suficiente substância* para que o gene u_5 se expresse.
t_{51}, t_{52}, t_{53} : limiar de substância para u_5.
n_{51}, n_{52}, n_{53} : osciladores de período aleatoriamente variável.

B.2.2 Gene P

$$\text{Se } \mathbf{P[t-1] \geq 0} \quad \Rightarrow \quad \mathbf{P = 0}$$

senão

		$\mathbf{S_{pfb}}$		
		≤ -6	$-5, -4, -3, -2$	≥ -1
$\mathbf{S_{fp}}$	≥ 5	**2**	**2**	**2**
	$3, 4$	**1**	**1**	**2**
	$1, 2$	**0**	**0**	**1**
	≤ 0	**0**	**0**	**0**

Valores de P quando $\mathbf{P[t-1] \leq 0}$

onde

$$S_{fp} = \sum_{i=1}^{5} FP[i]$$

FP: integração dos sinais de excitação de $u_1, \ldots, u_5$.

e

$$S_{pfb} = v_{fp}[1] + \sum_{i=1}^{2} w_{ifp} + \sum_{i=1}^{6} x_{ifp} + \sum_{i=1}^{2} y_{ifp} + z[1]$$

S_{pfb}: somatório dos sinais de realimentação das camadas de genes v, w, x, y e z.

$$FP = \begin{cases} -2 & \text{se} \quad SFu = -10, \ldots, -6 \\ -1 & \text{se} \quad SFu = -5, \ldots, -3 \\ 0 & \text{se} \quad SFu = -2, \ldots, 2 \\ 1 & \text{se} \quad SFu = 3, \ldots, 5 \\ 2 & \text{se} \quad SFu = 6, 10 \end{cases}$$

onde

$$SFu = \sum_{i=1}^{5} Fu_i[1]$$

e

$$Fu_i = \begin{cases} -1 & \text{se} \quad S_i = -10, \ldots, -4 \\ 0 & \text{se} \quad S_i = -3, \ldots, 0 \\ 1 & \text{se} \quad S_i = 1, \ldots, 4 \\ 2 & \text{se} \quad S_i = 5, \ldots, 10 \end{cases}$$

onde

$$S_i = \sum_{j=2}^{6} u_i[j]$$

B.2.3 Gene v

$$\mathbf{v} = \begin{cases} \mathbf{2} & \text{se} \quad Sp \geq 5 & \text{e} \quad v[t-1] = 0 \\ \mathbf{1} & \text{se} \quad 1 \leq Spf \leq 4 & \text{e} \quad v[t-1] = 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

onde

$$Sp = \sum_{i=1}^{5} P[i]$$

e

$$Spf = Sp + \frac{1}{2}\sum_{i=1}^{2}\sum_{i=j}^{2} w_i[j]$$

$$\mathbf{v_{fp}} = \begin{cases} \mathbf{-2} & \text{se} \quad Sp \geq 5 & \text{e} \quad v[t-1] = 0 \\ \mathbf{-1} & \text{se} \quad 1 \leq Spf \leq 4 & \text{e} \quad v[t-1] = 0 \\ \mathbf{0} & \text{caso contrário} & \end{cases}$$

com Sp e Spf como antes.

B.2.4 Genes w

Gene $\mathbf{w_1}$

$$\mathbf{w_1} = \begin{cases} \mathbf{2} & \text{se} \quad Sv \geq 5 & \text{e} \quad w_1[t-1] \leq 0 \\ \mathbf{1} & \text{se} \quad 1 \leq Svf_1 \leq 4 & \text{e} \quad w_1[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} & \end{cases}$$

onde

$$Sv = \sum_{i=1}^{5} v[i]$$

e

$$Svf_1 = Sv + \frac{1}{3}\sum_{i=1}^{3}\sum_{j=1}^{2} x_i[j]$$

$$\mathbf{w_{1f}} = \begin{cases} \mathbf{-2} & \text{se} \quad (Sv \geq 5 \quad \text{ou} \quad 1 \leq Svf_1 \leq 4) \\ & \text{senão} \\ \mathbf{-1} & \text{se} \quad w_1[t-1] > 0 \quad \text{ou} \quad w_1[t-2] > 0 \quad \text{ou} \quad w_1[t-3] > 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Sv e Svf_1 como antes.)

$$\mathbf{w_{1fp}} = \begin{cases} \mathbf{-2} & \text{se} \quad Sv \geq 5 & \text{e} \quad w_1[t-1] \leq 0 \\ \mathbf{-1} & \text{se} \quad 1 \leq Svf_1 \leq 4 & \text{e} \quad w_1[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} & \end{cases}$$

(Sv e Svf_1 como antes.)

Gene $\mathbf{w_2}$

$$\mathbf{w_2} = \begin{cases} \mathbf{2} & \text{se} \quad Sv \geq 5 & \text{e} \quad w_2[t-1] \leq 0 \\ \mathbf{1} & \text{se} \quad 1 \leq Svf_2 \leq 4 & \text{e} \quad w_2[t-1] = 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

onde

$$Sv = \sum_{i=1}^{5} v[i]$$

e

$$Svf_2 = Sv + \frac{1}{3}\sum_{i=4}^{6}\sum_{j=1}^{2} x_i[j]$$

$$\mathbf{w_{2f}} = \begin{cases} \mathbf{-2} & \text{se} & (Sv \geq 5 & \text{ou} & 1 \leq Svf_2 \leq 4) \\ & \text{senão} & & & \\ \mathbf{-1} & \text{se} & w_2[t-1] > 0 & \text{ou} \quad w_2[t-2] > 0 & \text{ou} \quad w_2[t-3] > 0 \\ \mathbf{0} & & \text{caso contrário} & & \end{cases}$$

(Sv e Svf_2 como antes.)

$$\mathbf{w_{2fp}} = \begin{cases} \mathbf{-2} & \text{se} \quad Sv \geq 5 & \text{e} \quad w_2[t-1] \leq 0 \\ \mathbf{-1} & \text{se} \quad 1 \leq Svf_2 \leq 4 & \text{e} \quad w_2[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Sv e Svf_2 como antes.)

B.2.5 Genes x

Gene $\mathbf{x_i}$, (para cada $i = 1, 2, 3$ –fixo–)

$$\mathbf{x_i} = \begin{cases} \mathbf{2} & \text{se} \quad Sw_1 \geq 5 & \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{1} & \text{se} \quad 1 \leq Swf_1 \leq 4 & \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

onde

$$Sw_1 = \sum_{j=1}^{5} w_1[j]$$

e

$$Swf_1 = Sw_1 + \sum_{j=1}^{2} y_{1f}[j]$$

$$\mathbf{x_{if}} = \begin{cases} \mathbf{-2} & \text{se} \quad (Sw_1 \geq 5 \quad \text{ou} \quad 1 \leq Swf_1 \leq 4) \\ & \text{senão} \\ \mathbf{-1} & \text{se} \quad x_i[t-1] > 0 \quad \text{ou} \quad x_i[t-2] > 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Sw_1 e Swf_1 como antes.)

$$\mathbf{x_{ifp}} = \begin{cases} \mathbf{-2} & \text{se} \quad Sw_1 \geq 5 \quad \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{-1} & \text{se} \quad 1 \leq Swf_1 \leq 4 \quad \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Sw_1 e Swf_1 como antes.)

Gene $\mathbf{x_i}$, (para cada $i = 4, 5, 6$ –fixo–)

$$\mathbf{x_i} = \begin{cases} \mathbf{2} & \text{se} \quad Sw_2 \geq 5 \quad \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{1} & \text{se} \quad 1 \leq Swf_2 \leq 4 \quad \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

onde

$$Sw_2 = \sum_{j=1}^{5} w_2[j]$$

e

$$Swf_2 = Sw_2 + \sum_{j=1}^{2} y_{2f}[j]$$

$$\mathbf{x_{if}} = \begin{cases} \mathbf{-2} & \text{se} \quad (Sw_2 \geq 5 \quad \text{ou} \quad 1 \leq Swf_2 \leq 4) \\ & \text{senão} \\ \mathbf{-1} & \text{se} \quad x_i[t-1] > 0 \quad \text{ou} \quad x_i[t-2] > 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Sw_2 e Swf_2 como antes.)

$$\mathbf{x_{ifp}} = \begin{cases} \mathbf{-2} & \text{se} \quad Sw_2 \geq 5 & \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{-1} & \text{se} \quad 1 \leq Swf_2 \leq 4 & \text{e} \quad x_i[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} & \end{cases}$$

(Sw_2 e Swf_2 como antes.)

B.2.6 Genes y

Gene $\mathbf{y_1}$

$$\mathbf{y_1} = \begin{cases} \mathbf{Mx_1} & \text{se } y_1[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

onde

$$Mx_1 = \frac{1}{3}\sum_{i=1}^{3} x_i[1]$$

$$\mathbf{y_{1f}} = \begin{cases} \mathbf{-2} & \text{se } (Mx_1 \geq 0 \text{ e } y_1[t-1] \leq 0) \\ & \text{senão} \\ \mathbf{-1} & \text{se } y_1[t-1] > 0 \text{ ou } y_1[t-2] > 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Mx_1 como antes.)

$$\mathbf{y_{1fp}} = \begin{cases} \mathbf{-Mx_1} & \text{se } y_1[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Mx_1 como antes.)

Gene $\mathbf{y_2}$

$$\mathbf{y_2} = \begin{cases} \mathbf{Mx_2} & \text{se } y_2[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

onde

$$Mx_2 = \frac{1}{3}\sum_{i=4}^{6} x_i[1]$$

$$\mathbf{y_{2f}} = \begin{cases} \mathbf{-2} & \text{se } (Mx_2 \geq 0 \text{ e } y_2[t-1] \leq 0) \\ & \text{senão} \\ \mathbf{-1} & \text{se } y_2[t-1] > 0 \text{ ou } y_2[t-2] > 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Mx_2 como antes.)

$$\mathbf{y_{2fp}} = \begin{cases} \mathbf{-Mx_2} & \text{se } y_2[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

(Mx_2 como antes.)

B.2.7 Gene z

$$\mathbf{Z} = \begin{cases} \frac{\mathbf{y_1[t-1]+y_1[t-1]}}{\mathbf{2}} & \text{se} \quad z[t-1] \leq 0 \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

Apêndice C

Árvores de Classificação por Multirresolução

Os classificadores binários podem reconhecer se um objeto (configuração de valores das variáveis de entrada) pertence ou não a uma determinada classe atribuindo um rótulo de classificação em $\{0, 1\}$. Assim, se existem n classes diferentes, a classificação pode ser realizada por uma familia de n operadores binários, um para o reconhecimento dos objetos em cada uma das classes. Os multiclassificadores são uma extensão natural dos classificadores binários [24]. Neles, um operador associa um rótulo de classificação em $\{0, 1, \ldots, n\}$ a cada uma das configurações de variáveis de entrada observadas.

As árvores de classificação dividem esta tarefa em passos sucessivos de multiclassificação, dividindo a família de classes em sub-famílias, a fim de obter uma maior precisão nos primeiros passos onde resulta mais fácil a discriminação. Assim, os objetos são observados em diversas resoluções. Neste trabalho se propõe uma técnica automática para gerar as árvores de classificação e especificar como deve ser feito o treinamento dos operadores de multiclassificação correspondentes [29]. Esta técnica de aprendizado foi avaliada no reconhecimento óptico de caracteres, mas ela pode ser utilizada para resolver outros pro-

blemas de classificação, como por exemplo, a identificação das componentes das funções de transição de um sistema dinâmico finito.

Um sistema de OCR gera documentos eletrônicos (arquivos ASCII) correspondentes a materiais impressos tais como livros e outros tipos de documentos, em geral mediante o uso de procedimentos de análise de imagens.

O enfoque proposto em nosso método utilizou operadores morfológicos para o reconhecimento de caracteres, estendendo um resultado prévio sobre o uso de árvores de classificação em OCR [24]. Naquele trabalho, foi apresentado um método baseado em uma árvore de classificação. Em lugar de projetar um classificador que discrimina todos os objetos em um único passo de classificação, o processo é executado como um procedimento multi-etapas, isto é, os objetos são classificados em sub-classes e depois cada sub-classe é novamente classificada em mais sub-classes. No entanto, a parte mais difícil da técnica proposta era o projeto da árvore de classificação. A inovação apresentada neste trabalho consiste em uma técnica automática para o projeto de árvores de classificação.

C.1 Operadores Morfológicos e Multiclassificadores

Uma imagem se compõe habitualmente de vários objetos, e cada objeto pode corresponder a mais de um componente na imagem. O propósito da classificação é associar um código de classificação único a cada objeto na imagem, que indica a classe à qual ele pertence. Isto pode ser feito associando o código de classificação correto a todos os pixels na imagem.

Seja $E = Z^2$ o plano inteiro e $\mathcal{P}(E)$ o conjunto de partes de E. Uma imagem binária em E pode ser modelada por uma função binária $f : E \to \{0, 1\}$, ou equivalentemente,

por um subconjunto $S \subseteq E$, fazendo $x \in S \iff f(x) = 1$, para todo $x \in E$.

Podem-se usar W-operadores binários –que são mapeamentos do tipo $\Psi : \mathcal{P}(E) \to \mathcal{P}(E)$ invariantes por translação e localmente definidos numa janela W [29]– , ajustando o tamanho da janela, para reconhecer formas de objetos que pertencem a uma classe particular. Assim, se existem n classes diferentes, a classificação pode ser realizada por uma familia ψ_j de n operadores binários, um para o reconhecimento dos objetos em cada uma das classes. A desvantagem desta estratégia é que é preciso projetar um operador para cada classe. Também temos que considerar como tratar os objetos que são reconhecidos por mais de um operador.

Quando existem n classes diferentes é mais conveniente o uso de um multiclassificador ao invés de n classificadores binários. Seja $\{0, 1, \ldots, n\}^E$ o conjunto de todos os mapeamentos de E em $\{0, 1, \ldots, n\}$ (em nosso contexto correspondem ao conjunto de imagens em níveis de cinza definidas em E, com $n + 1$ níveis de cinza) e consideremos um operador da forma $\Psi : \mathcal{P}(E) \to \{0, 1, \ldots, n\}^E$, caracterizado por uma função multi-nível $\psi : \mathcal{P}(W) \to \{0, 1, \ldots, n\}$. Aqui, um operador associa um rótulo de classificação em $\{0, 1, \ldots, n\}$ a cada uma das formas em W. Neste caso, a entrada é uma imagem binária e a saída é uma imagem classificada em níveis de cinza. Cada pixel na imagem de saída tem um valor entre 0 e n, o rótulo de classificação. O rótulo 0 é reservado para o fundo, e n indica o número de classes consideradas.

A vantagem do uso de multi-classificadores está no fato de que em lugar de projetar n operadores, um para cada classe, só é preciso projetar um operador.

Um multiclassificador deste tipo, pode ser representado pelos *kernels de ψ nos níveis i* ou equivalentemente pelas *bases de ψ nos níveis i*, onde $0 \leq i \leq n$ [29].

Os multiclassificadores são construídos a partir dos dados de treinamento, que neste caso são um conjunto de pares de imagens observadas-ideais, que ilustram a classificação. As imagens observadas são imagens binárias contendo os objetos a serem classificados e as imagens ideais são as respectivas imagens com o rótulo de classificação correto.

Para cada forma $\mathbf{x} \subseteq W$, a classe c, à qual $\mathbf{x}$ pertence, é definida pelo rótulo mais freqüentemente atribuído nos dados de treinamento. Depois, para cada forma $\mathbf{x}$, se projeta um operador ψ tal que $\psi(\mathbf{x}) = c(\mathbf{x})$, obtendo cada uma das n bases, correspondentes aos intervalos maximais do kernel [29]. Neste procedimento usamos o algoritmo ISI [31].

A construção de multi-classificadores pode ser ineficiente computacionalmente se o número de classes e o tamanho da janela são grandes. Outro problema é a precisão na estimação das probabilidades condicionais. Recentemente, as árvores de classificação foram propostas como una estratégia para superar estas dificuldades.

C.2 Árvores de Classificação

Uma *árvore de classificação* divide a tarefa original de classificação em vários passos de classificação. Em lugar de atribuir a classificação correta num único passo de classificação, os objetos são inicialmente classificados em sub-famílias. Depois, no segundo passo da classificação, cada uma das sub-famílias são classificadas em outro número de sub-famílias, e este processo é repetido para cada sub-família até que a classificação completa é realizada. Esta é mostrada no diagrama da Figura C.1.

No exemplo da figura C.1, o processo da classificação consiste de três passos. O conjunto total de classes ($\{0, 1, 2, 3, 4, 5, 6, 7\}$) é primeiramente dividido em três sub-famílias,

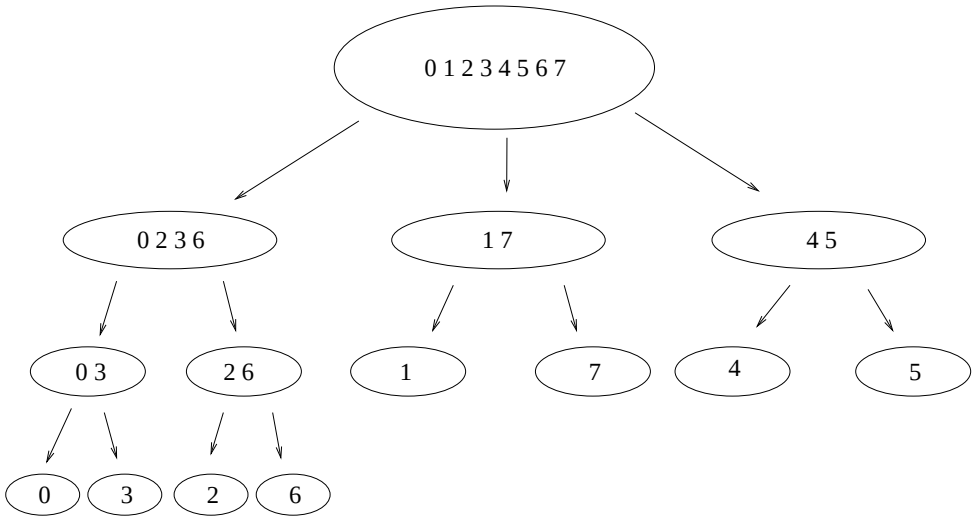


Figura C.1: Exemplo de uma Árvore de Classificação.

os conjuntos $\{0, 2, 3, 6\}$, $\{1, 7\}$ e $\{4, 5\}$. O conjunto $\{0, 2, 3, 6\}$ é depois dividido em dois subconjuntos, $\{0, 3\}$ e $\{2, 6\}$, enquanto os outros dois são divididos em conjuntos unitários, e assim por diante.

Cada nó não folha V de uma árvore de classificação contém a seguinte informação :

- uma sub-família $\mathcal{F}(V)$ de classes.
- uma lista de nós filhos $child(V)$. As sub-famílias associadas aos nós formam uma partição disjunta de $\mathcal{F}(V)$.
- um operador (multi-classificador) Ψ_V que classifica os objetos contidos em classes de $\mathcal{F}(V)$ nas respectivas sub-famílias dos seus nós filhos.

Os nós folha contém objetos de uma única classe. Por tanto, não existe operador associado aos nós folha.

Uma vez que tal estrutura é dada, necessitamos projetar operadores capazes de clas-

sificar os objetos de acordo com a árvore. Por exemplo, o operador associado ao nó raiz deve poder separar a família de todos os objetos possíveis ($\{0,1,2,3,4,5,6,7\}$) em três subconjuntos $\{0,2,3,6\}$, $\{1,7\}$ e $\{4,5\}$. O operador associado ao primeiro nó deve poder separar os objetos de $\{0,2,3,6\}$ em duas sub-famílias: $\{0,3\}$ e $\{2,6\}$, e assim por diante.

Dada uma imagem e a árvore de classificacão cujo conjunto de objetos no nó raiz é o conjunto de todos os objetos presentes na imagem, para classificar estes objetos, deve-se atravessar a árvore inteira (exceto os nós folha) aplicando os operadores. O operador do nó raiz é aplicado à imagem original. Uma imagem é criada para cada nó filho, contendo somente aqueles objetos classificados como sendo da respectiva sub-família. Logo, o operador de cada um dos nós filhos é aplicado nas respectivas imagens, e assim por diante, até que um nó folha é alcançado. O resultado da classificação é obtido a partir de todas as imagens finais.

A idéia básica desta técnica é que a classificação dos objetos em classes intermediárias deve resultar mais fácil porque o operador não necessita realizar discriminações complexas de formas.

C.3 Projeto de Árvores de Classificação

Para projetar uma árvore de classificação, usualmente é fixada uma estrutura de árvore e depois é projetado um operador para cada nó não folha usando o procedimento de treinamento indicado na Seção C.1. O tamanho da janela é testado experimentalmente para cada nó, até que um resultado aceitável é encontrado. Se um caracter recebe mais do que uma certa porcentagem de classificação como pertencendo a uma sub-família, então ele é mantido nas imagens de treinamento do nó filho respectivo. Devido a isto,

um caracter pode estar em imagens de mais que um dos seus filhos. Se um caracter é mantido numa sub-família onde ele não deveria estar, o operador projetado para dividir essa sub-família tende a eliminá-lo porque ele é treinado para reconhecer somente aqueles caracteres presentes na respectiva sub-família.

Uma parte difícil no desenho de árvores de classificação é especificar a família de classes associada a cada nó. No desenho de árvores de classificação, uma escolha natural é manter os objetos que tem formas similares na mesma sub-família.

A nova técnica proposta desenha automaticamente uma estrutura de árvore de classificação. O objetivo é usar janelas pequenas nos primeiros níveis da árvore e gerar grupos intermediários que possam ser facilmente discriminados.

C.3.1 Matriz de classificação

Seja $\mathcal{F}$ uma família de classes de objetos. Queremos encontrar uma janela W relativamente pequena que seja suficiente para discriminar objetos em diferentes classes de $\mathcal{F}$. Para analisar o poder discriminatório de uma janela W dada, introduzimos a *matriz de classificação*.

Na construção de uma matriz de classificação, para uma janela W e uma família $\mathcal{F}$ dadas, desenha-se um multi-classificador baseado em W a partir de imagens contendo objetos das classes em $\mathcal{F}$, de acordo com a descrição feita na Seção C.1. Este operador é aplicado às mesmas imagens, para calcular $T(i, j)$, i.e., quantos pixels de caracteres da classe i são classificados como sendo da classe j. A soma de uma linha i em T indica o número total de pixels correspondentes a objetos da classe i.

A Figura C.2 mostra uma matriz de classificação para a família $\{A, B, C, D, E, F\}$.

Nesta matriz podemos ver que 210 pixels do caracter A são corretamente classificados

T(i,j)	A	B	C	D	E	F
A	210	0	0	0	0	3
B	0	239	0	43	0	0
C	0	0	172	0	0	0
D	0	0	0	381	0	0
E	0	0	0	0	118	8
F	2	0	0	0	28	271

Figura C.2: Exemplo de uma Matriz de Classificação.

enquanto 3 deles são incorreta-mente classificados como sendo do caracter F. Dois pixel do caracter F são classificados como sendo do caracter A e 28 pixels como do caracter E, sobre 271 classificações corretas. Por outro lado, nenhuma parte dos caracteres A, E e F tem sido classificados como B, C ou D, e nenhuma parte destas letras como A, E ou F. Assim, podemos dizer que este classificador é capaz de separar a família $\{A, E, F\}$ da família $\{B, C, D\}$. Ainda mais, podemos ver que a letra C nunca é mal classificada, e nenhuma outra letra é mal classificada como letra C. Então, podemos particionar a família $\{A, B, C, D, E\}$ em três subconjuntos, $S_1 = \{A, E, F\}$, $S_2 = \{B, D\}$ e $S_3 = \{C\}$.

C.3.2 Grafo de Semelhança

O poder discriminador de um multiclassificador pode ser analisado a través de sua matriz de classificação. Para analisar a semelhança entre objetos de distintas classes, definimos um grafo não orientado G, onde os vértices são as classes (caracteres), e existe uma aresta entre dois vértices i e j se, e somente se, $T(i, j) > 0$ ou $T(j, i) > 0$. Este grafo, induzido pela matriz de classificação, é o *grafo de semelhança*, onde caracteres parecidos são conectados por uma aresta. Neste grafo, podem acontecer três casos:

1. O grafo é completamente desconectado, i.e., $T(i,j) = 0, \forall i,j,\ i \neq j$. Este é o caso ideal, i.e., a janela é suficiente para discriminar todos os caracteres.

2. O grafo é totalmente conectado, i.e., $T(i,j) > 0, \forall i,j,\ i \neq j$. Isto indica que os caracteres contém formas parecidas em relação à janela W e não podem ser discriminados; a janela W é muito pequena. Quanto maiores são os valores de $T(i,j)$ e $T(j,i)$, maior é a semelhança entre os caracteres i e j.

3. O grafo contém sub-grafos conectados não triviais. Neste caso, o operador não classifica todos os objetos corretamente. É possível, no entanto, agrupar os caracteres em sub-famílias de uma forma tal que nenhuma parte de um caracter numa sub-família recebe a classificação de um caracter em outra sub-família.

A Figura C.3 mostra o grafo obtido da matriz T da Figura C.2. Podemos ver que existem três sub-grafos conectados com conjuntos de vértices $V_1 = \{AFE\}$, $V_2 = \{BD\}$ e$V_3 = \{C\}$, respectivamente.

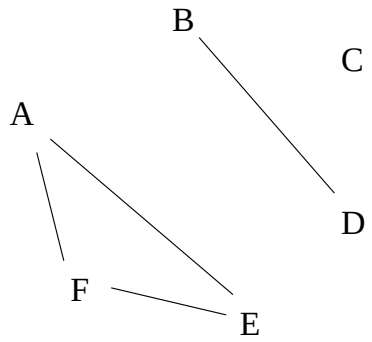


Figura C.3: Subgrafos Conectados.

Isto significa que um operador baseado em W poderia discriminar os elementos em V_1 dos elementos em V_2 ou V_3. No entanto, ele não poderia discriminar claramente os

caracteres A, F e E, porque existe, por exemplo, alguma "confusão" entre as formas presentes em A e F. Nossa conclusão aqui é que o operador baseado em W poderia ser usado para dividir o conjunto $\{A, B, C, D, E, F\}$ em três subconjuntos.

Em geral, o caso (3) resulta numa família $\{G_1, G_2, \ldots, G_p\}$ de sub-grafos disjuntos, o que significa que nenhuma parte de um objeto na classe de G_i tem sido classificada como sendo de uma classe em outro sub-grafo G_j. Neste caso, a janela é capaz de separar elementos de sub-grafos diferentes. Portanto, se definimos os conjuntos $V_1, V_2, \ldots, V_p$ como as sub-famílias de um nó, um W-operador seria capaz de discriminar os elementos de G nas p sub-famílias.

Na prática, mesmo que as classificações erradas $T(i,j)$ e $T(j,i)$ não sejam zero, estes valores podem ser muito pequenos, comparados ao número de pixels correspondentes aos caracteres i e j. Em outras palavras, se somente uma pequena parte de um caracter é classificada incorreta-mente, ainda é possível atribuir a classificação certa a ele.

Sejam $N(i)$ e $N(j)$ o número de pixels de todas as ocorrências dos caracteres i e j, respectivamente, definimos

$$N(i,j) = \min\{N(i), N(j)\}\,. \tag{C.1}$$

O grafo G é criado de uma matriz de classificação levemente modificada, que denotaremos por T', obtida considerando um limiar que depende de $N(i,j)$ e do parâmetro percentual θ, $0 \leq \theta \leq 100$:

$$T'(i,j) = \begin{cases} T(i,j), & i \neq j \text{ and } T(i,j) > \frac{\theta \cdot N(i,j)}{100} \\ 0, & \text{otherwise.} \end{cases} \tag{C.2}$$

O grafo resultante terá uma aresta entre duas classes se, e somente se, a classsificação errada entre eles é significativa, i.e., se o número de pixels mal classificados entre eles é uma percentagem considerável (definida por θ) do número de pixel pertencentes a essas classes. Isto pode causar alguma perda na qualidade do classificador, mas tenta evitar o uso de janelas grandes, que não são desejáveis porque requerem uma grande quantidade de dados de treinamento para alcançar uma boa precisão. A perda na qualidade da classificação pode ser superada pelo ganho em precisão.

C.3.3 Projeto Automático da Árvore de Classificação

Para definir uma estrutura de árvore de classificação, cria-se inicialmente um nó raiz e constrói-se um grafo G a partir da matriz de classificação T', obtida com a família de todas as classes $\{0, 1, \ldots, n\}$ e usando a primeira janela na seqüência de janelas crescentes. Em nossos experimentos usamos a seqüência 3×3, 5×3, 5×5, 7×5,... (Figura C.4).

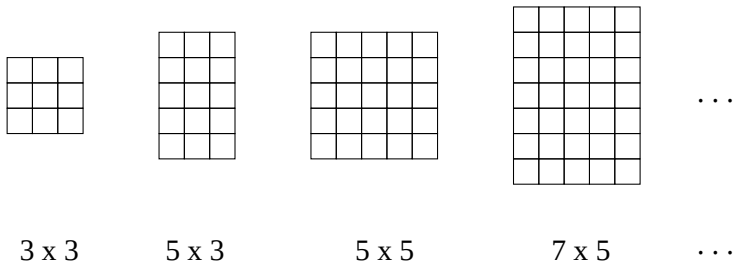


Figura C.4: Seqüência de janelas.

O procedimento sucessivamente constrói uma árvore de classificação de acordo com as regras descritas a seguir. Estas regras dependem do caso que ocorre em G (listado no começo desta seção).

Se ocorre o caso (1), o processo termina para o nó correspondente.

Se o caso (2) ocorre, significa que a janela é muito pequena. Logo, o processo é repetido para esse nó, usando a próxima janela na seqüência de janelas. Dependendo da forma dos caracteres, pode acontecer que até janelas muito grandes não consigam separar caracteres em conjuntos disjuntos. Por outro lado, não é desejável permitir que o tamanho da janela aumente demais, porque isso implicaria numa precisão muito pobre para o operador treinado. Para evitar este problema, cada vez que a janela é aumentada, o parâmetro θ também é aumentado por um fator dado. Isto pode implicar que a qualidade da discriminação poderia ser comprometida, mas como isto acontecerá a um ramo da árvore, pode resultar insignificante em relação ao resultado geral da classificação.

Finalmente, se ocorre o caso (3), a família de classes nesse nó é particionada em sub-famílias correspondentes aos sub-grafos disjuntos. Para cada sub-família é criado um nó filho e o processo termina para esse nó. A janela a ser usada para os nós filhos é a próxima na seqüência e o parâmetro θ não é alterado.

O processo é repetido iterativamente enquanto existem nós a serem processados. Ao final deste processo, fica definida uma estrutura de árvore com uma janela e uma família de classes associada a cada nó. Uma vez que a estrutura está definida, o próximo passo consiste em treinar um multi-classificador para cada nó. Como vimos antes, cada multi-classificador é treinado para separar os caracteres na família de classes associada a esse nó em sub-famílias correspondentes aos seus nós filhos.

A Figura C.5 mostra uma árvore de classificação completa, exceto pelos nós folha, desenhada para o problema de reconhecimento de caracteres discutido na Seção C.4.1. A Figura C.6 mostra em detalhe um ramo desta árvore correspondente à subfamília composta pelos caracteres "mr1IliLY". Uma janela 7×7 é suficiente para discriminar os

caracteres "m" e "r" dos outros. Uma janela maior, 9×9, reconhece os caracteres "L" e "Y", enquanto uma janela 11×9 reconhece os caracteres "T" e "i". A última classe, contem três caracteres muito parecidos ("1", "I" e "l") e foi necessário aumentar a janela até 13×11. Usualmente, para uma janela deste tamanho, a quantidade de dados de treinamento requerida para uma boa precisão é muito grande. Neste caso, este problema é atenuado porque a variedade de formas a serem analisadas é limitada.

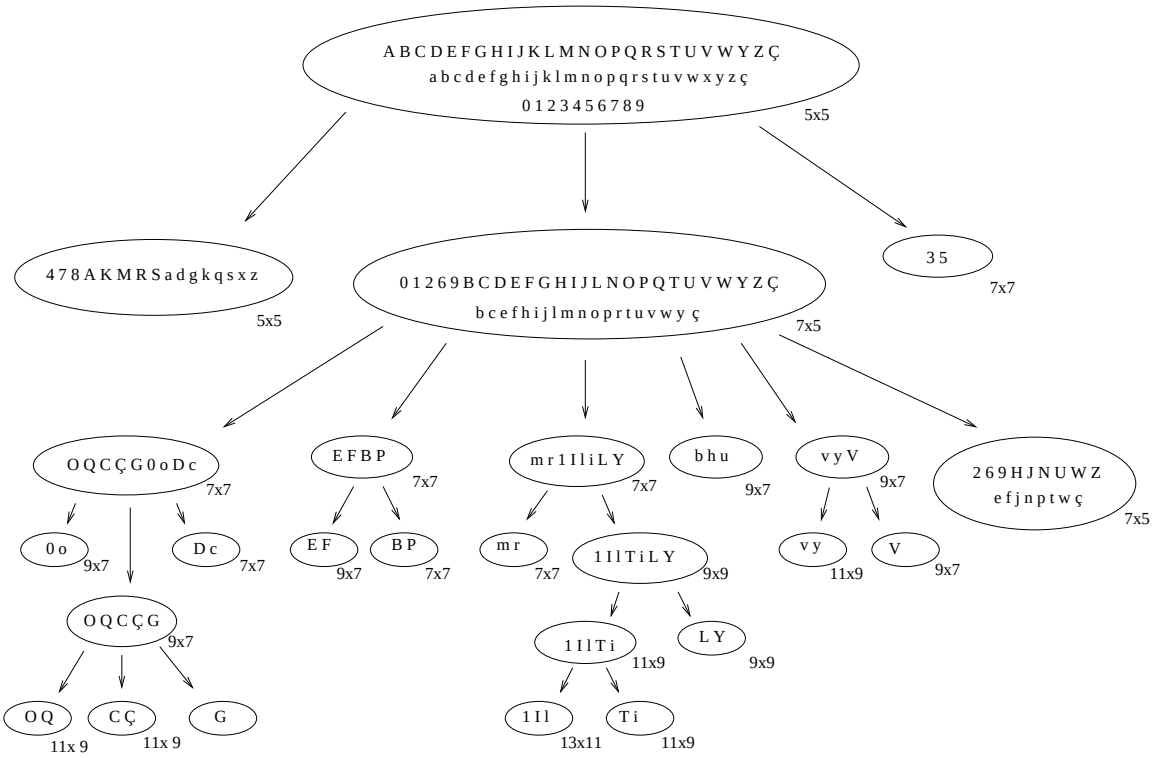


Figura C.5: Um exemplo de árvore de classificação.

C.4 Resultados Experimentais

As duas partes principais do OCR implementado são as componentes de *treinamento* e *aplicação*. A componente de treinamento permite ao usuário treinar uma árvore de clas-

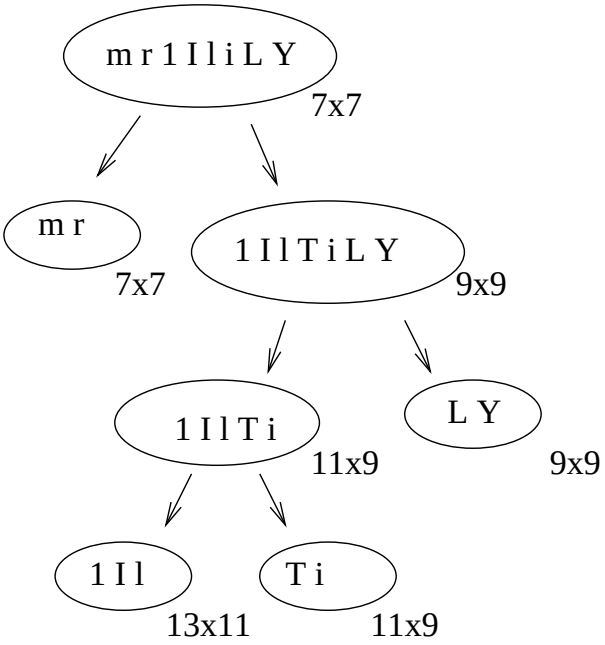


Figura C.6: Uma sub-árvore da árvore de classificação.

sificação a partir de amostras de imagens observadas-ideais. A componente de aplicação permite ao usuário aplicar uma árvore de classificação a imagens observadas diferentes das usadas no treinamento e obter a classificação correspondente.

A nova metodologia para o projeto de árvores de classificação foi testada para dois grupos de imagens: um conjunto de imagens obtidas de materiais impressos com impressora laser corrompidos com ruido de bordas, e outro conjunto de imagens obtidos de documentos impressos por uma impressora de matriz de pontos.

C.4.1 Imagens Digitalizadas de Impressora Laser

As imagens originais eram compostas de 63 caracteres diferentes, as letras em maiúsculas e minúsculas, e os números de 0 a 9. Cada imagem consta de 630 caracteres, contendo exatamente 10 de cada caracter. Estas imagens foram obtidas escaneando texto gerado

por impressora laser a 300 dpi. Devido ao fato destas imagens serem de boa qualidade, adicionamos 25% de ruído sal e pimenta nas bordas externas dos caracteres, com o objetivo de simular uma degradação que pode ocorrer usualmente em imagens de texto escaneado.

A árvore de classificação obtida foi testada em 10 diferentes imagens do mesmo tipo. A Tabela C.1 mostra o resultado destes testes, incluindo também testes feitos com algums OCRs comerciais. Comparado com estes OCRs comerciais, o desempenho de nosso OCR foi significativamente melhor. Em cada imagem, de um total de 630 caracteres, somente um pequeno número deles (entre 7 e 13) foram classificados erroneamente.

OCR	Precisão
Textbridge	87.89%
Cuneiform'99	90.63%
Nosso	98.33%

Tabela C.1: Precisão no primeiro grupo de imagens.

C.4.2 Imagens Digitalizadas de Impressora de Matriz de Pontos

As imagens de matriz de pontos geram imagens de qualidade muito pobre, porque as letras estão formadas a grosso modo por pontos.

O resultado dos testes de classificação efetuados se mostra na Tabela C.2 junto com os resultados de um OCR comercial. O desempenho do OCR comercial nas imagens originais de níveis de cinza é muito pobre. Então, também testamos ele com images filtradas por um limiar seguido de dois filtros de mediana e também por nosso pre-processamento. Neste OCR comercial, a opcão "dot matrix" estaba habilitada no processamento das imagens.

O melhor resultado do OCR comercial, o qual se deve a uma filtragem externa, fica estatisticamente perto de nosso resultado. No entanto, nosso resultado pode ser melhorado

OCR	Imagem testada	Precisão
Cuneiform'99	Original Níveis de Cinza	20.19%
Cuneiform'99	Nosso pre-processamento	77.83%
Cuneiform'99	Limiar + Medianas	85.97%
Nosso	Nosso pre-processamento	88.58%

Tabela C.2: Precisão no segundo grupo de imagens.

usando mais dados de treinamento, já que somente quatro imagens de treinamento tem sido usadas neste experimento.

C.5 Conclusões

A técnica apresentada para a geração automática de árvores de classificação, se baseia nos conceitos de *matriz de classificação* e *grafo de semelhança*. O método resulta *eficiente* porque não precisa analisar todas as possíveis combinações de subfamílias para uma dada família de classes. As subfamílias são automaticamente geradas analisando o grafo de semelhança, obtido da matriz de classificação que, essencialmente, expressa quão bem uma janela W discrimina os objetos em diferentes classes.

Uma característica desta técnica é que sempre tenta encontrar a *menor janela* suficiente para separar o conjunto de objetos em subconjuntos disjuntos. Isto permite que o mesmo objeto seja analisado em *diferentes resoluções*. Basicamente, nos passos iniciais da classificação, é possível fazer uma separação grosseira de caracteres com uma janela relativamente pequena. Nos passos posteriores do processo de classificação, a janela requerida para separar os caracteres é maior, no entanto, a variedade de objetos que se precisa reconhecer é usualmente muito pequena, fazendo a classificação estatisticamente mais fácil.

Os conceitos apresentados aqui têm sido utilizados no contexto de projeto de OCR, embora eles possam ser aplicados a outros problemas de classificação, como por exemplo o aprendizado das componentes da função de transição na identificação de um sistema dinâmico finito. Os resultados obtidos para dois conjuntos de imagens são melhores que os de OCRs comerciais. Devido a que nossa técnica é baseada em treinamento, é possível criar com ela *classificadores especialmente adaptados* a problemas específicos de classificação.

Referências Bibliográficas

[1] D. J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J. M. Trent. Expression Profiling Using cDNA Microarrays. *Nature (Genetics Supplement)*, 21:10–14, January 1999.

[2] J. Barrera, E. R. Dougherty, M. D. Gubitoso, and N. S. T. Hirata. Modeling Temporal Morphological Systems via Lattice Dynamical Systems. In *Nonlinear Image Processing and Pattern Analysis XII, Proceedings of SPIE Vol. 4304*, pages 265–276, 2001.

[3] P. D'Haeseleer, S. Liang, and R. Somogyi. Gene Expression Analysis and Genetic Network Modeling. PSB'99 Tutorial.

[4] P. O. Brown. Exploring the New World of the Genome with DNA Microarrays. *Nature (Genetics Supplement)*, 21:33–37, January 1999.

[5] J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale. *Science*, 278(24):680–686, October 1997.

[6] D. J. Lockhart and E. A. Winzeler. Genomics, gene expression and DNA arrays. *Nature*, 405:827–836, June 2000.

[7] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[8] T. M. Mitchell. *Machine Learning.* McGraw-Hill Series in Computer Science. McGraw-Hill, March 1997.

[9] F. J. Hill and G. R. Peterson. *Introduction to Switching Theory and Logical Design.* John Wiley, 3rd edition, 1981.

[10] R. H. Katz. *Contemporary Logic Design.* Benjamin Cummings, 1994.

[11] G. De Micheli. *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, 1994.

[12] N. S. Tomita. Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC. Master's thesis, Instituto de Matemática e Estatística - Universidade de São Paulo, São Paulo, SP - Brasil, março 1996.

[13] M. Anthony and N. Biggs. *Computational Learning Theory - An Introduction*. Cambridge University Press, 1992.

[14] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[15] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[16] D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100:78–150, 1992.

[17] G. J. F. Banon and J. Barrera. Decomposition of Mappings between Complete Lattices by Mathematical Morphology, Part I. General Lattices. *Signal Processing*, 30:299–327, 1993.

[18] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, Rhode Island, 3rd edition, 1967.

[19] N. S. Tomita. Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC. In *Anais do XXIII Seminário Integrado de Software e Hardware - IX Concurso de Teses e Dissertações - XV Concurso de Trabalhos de Iniciação Científica*, pages 517–525, Recife, PE, Agosto 1996. XVI Congresso da SBC, Sociedade Brasileira de Computação.

[20] H. A. Armelin, J. Barrera, E. R. Dougherty, J. E. Ferreira, M. D. Gubitoso, N. S. T. Hirata, and E. J. Neves. Simulator for Gene Expression Networks. In *Microarrays: Optical Technologies and Informatics, Proceedings of SPIE Vol. 4266*, pages 248–259, 2001.

[21] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[22] G. J. F. Banon and J. Barrera. Bases da Morfologia Matemática para Análise de Imagens Binárias. IX Escola de Computação, Pernambuco, Julho 1994.

[23] G. J. F. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM J. Applied Mathematics*, 51(6):1782–1798, December 1991.

[24] J. Barrera, R. Terada, R. A. Lotufo, N. S. T. Hirata, R. Hirata Jr., and F. A. Zampirolli. An OCR based on Mathematical Morphology. In *Nonlinear Image Processing IX*, volume 3304 of *Proceedings of SPIE*, pages 197–208, San Jose, CA, January 1998.

[25] R. Hirata Jr. *Projeto de Operadores Morfológicos – Restrições no Espaço e na Escala*. PhD thesis, Instituto de Matemática e Estatística, Universidade de São Paulo, December 2001.

[26] E. R. Dougherty and J. Barrera. Logical Image Operators. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Filters for Image Processing*, pages 1–60. SPIE and IEEE Press, Bellingham, 1999.

[27] S. Liang, S. Fuhrman, and R. Somogyi. REVEAL, A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures. In *PSB Proceedings*, volume 3, pages 18–29, 1998.

[28] A. J. Butte and I. S. Kohane. Mutual Information Relevance Networks: Functional Genomic Clustering Using Pairwise Entropy Measurements. In *PSB Proceedings*, volume 5, pages 415–426, 2000.

[29] Marcel Brun, Junior Barrera, Nina S. T. Hirata, Nestor W. Trepode, Daniel Dantas, and Routo Terada. Multi-resolution Classification Trees in OCR Design. In D. L. Borges and S.-T. Wu, editors, *Proceedings of Sibgrapi 2001*, pages 59–66, Florianopolis, Brasil, October 2001. IEEE.

[30] J. Barrera, P. A. Martin, E. R. Dougherty, M. D. Gubitoso, N. S. T. Hirata, and N. W. Trepode. Identification of Input-free Finite Lattice Dynamical Systems under Envelope Constraints. In *To appear in ISMM International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing, Sydney, Australia, April 2002.*

[31] J. Barrera, E. R. Dougherty, and N. S. Tomita. Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory. *Electronic Imaging*, 6(1):54–67, January 1997.