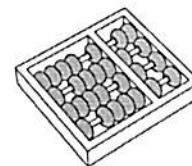


Jacqueline Midlej do Espírito Santo

**“Especificação e Detecção de Padrões Complexos de
Variáveis Ambientais em Aplicações de
Biodiversidade”**

CAMPINAS
2015



Universidade Estadual de Campinas
Instituto de Computação

Jacqueline Midlej do Espírito Santo

“Especificação e Detecção de Padrões Complexos de Variáveis Ambientais em Aplicações de Biodiversidade”

Orientador(a): Prof^a. Dr^a. Claudia Bauzer Medeiros

Dissertação de Mestrado apresentada ao Programa
de Pós-Graduação em Ciência da Computação do Instituto de Computação da
Universidade Estadual de Campinas para obtenção do título de Mestra em
Ciência da Computação.

ESTE EXEMPLAR CORRESPONDE À VERSÃO
DA DISSERTAÇÃO APRESENTADA À BANCA
EXAMINADORA POR JACQUELINE MIDLEJ
DO ESPÍRITO SANTO, SOB ORIENTAÇÃO DE
PROF^A. DR^A. CLAUDIA BAUZER MEDEI-
ROS.

A handwritten signature in blue ink, likely belonging to Claudia Bauzer Medeiros, the supervisor.

Assinatura do Orientador(a)

CAMPINAS

2015

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

Sa59e Santo, Jacqueline Midlej do Espírito, 1990-
Especificação e detecção de padrões complexos de variáveis ambientais em aplicações de biodiversidade / Jacqueline Midlej do Espírito Santo. – Campinas, SP : [s.n.], 2015.

Orientador: Claudia Maria Bauzer Medeiros.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Ciência da computação. 2. Banco de dados. 3. Processamento de eventos (Computação). 4. Biodiversidade. 5. Linguagem de consulta (Banco de dados). I. Medeiros, Claudia Maria Bauzer, 1954-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Complex pattern detection and specification for environmental variables in biodiversity applications

Palavras-chave em inglês:

Computer science

Databases

Event processing (Computer science)

Biodiversity

Query language (Databases)

Área de concentração: Ciência da Computação

Titulação: Mestra em Ciência da Computação

Banca examinadora:

Claudia Maria Bauzer Medeiros [Orientador]

André Santanchè

Hilda Carvalho de Oliveira

Data de defesa: 06-07-2015

Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado em Ciência da Computação, apresentada pelo(a) Mestrando(a) **Jacqueline Midlej do Espírito Santo**, aprovado(a) em **06 de julho de 2015**, pela Banca examinadora composta pelos Professores(as) Doutores(as):



Prof(a). Dr(a). Hilda Carvalho de Oliveira
Titular



Prof(a). Dr(a). André Santanchê
Titular



Prof(a). Dr(a). Claudia Maria Bauzer Medeiros
Presidente

Especificação e Detecção de Padrões Complexos de Variáveis Ambientais em Aplicações de Biodiversidade

Jacqueline Midlej do Espírito Santo¹

06 de julho de 2015

Banca Examinadora:

- Prof.^a. Dr.^a. Claudia Bauzer Medeiros (Orientadora)
- Prof. Dr. André Santanchè
Instituto de Computação - UNICAMP
- Prof.^a Dr.^a. Hilda Carvalho de Oliveira
Instituto de Geociências e Ciências Exatas - Unesp Rio Claro
- Prof.^a Dr.^a. Islene Calciolari Garcia
Instituto de Computação - UNICAMP (Suplente)
- Prof. Dr. David Corrêa Martins Junior
Centro de Matemática, Computação e Cognição - UFABC (Suplente externo)

¹Apoio Financeiro: Bolsa de estudos Fapesp (2013/02269-7) 2013–2015

Abstract

Biodiversity applications require a large variety of environmental data at multiple scales. It involves a huge amount of data from heterogeneous sources, in which sensor data streams are one of the main sources. An open problem in this context is how to specify and detect scenarios of interest from environmental variables, at multiple scales, to help scientists analyze phenomena and correlate results with data collected on the field. To help solve the problem, this dissertation is based on the theory of Complex Event Processing (CEP) to allow scientists to specify scenarios using patterns and detecting the occurrence of the scenario in real time. In the CEP literature, data are treated as events and patterns are described by event specifications and their relationships with each other. Event specification languages, however, do not support spatial aspects (which is essential in biodiversity applications) and neither do they contemplate complex composition operators. Given this context, this dissertation proposes a logic-based language to help scientists specify scenarios of interest. These scenarios are based in complex event composition. The main contributions are: proposal of a software architecture of a framework to detect complex events, extending the work of Koga [17]; a data model to represent events in biodiversity scenarios; and a language to specify event patterns in a hierarchical manner, exploring spatial and temporal relationships across events in many abstraction levels.

Resumo

Aplicações de biodiversidade se caracterizam por necessitar de uma grande variedade de dados ambientais em múltiplas escalas. Este contexto envolve uma enorme quantidade de dados gerados por fontes heterogêneas, sendo o fluxo de dados de sensores uma das principais fontes. Um problema em aberto neste contexto é como especificar e detectar cenários de interesse a partir de variáveis ambientais em múltiplas escalas, para facilitar aos cientistas a análise de fenômenos e correlações com dados coletados em campo. Para ajudar a solucionar o problema, a dissertação se baseia na teoria de Processamento de Eventos Complexos para permitir a especificação de cenários através de padrões e a detecção da ocorrência do cenário em tempo real. Nesta literatura, dados são tratados como eventos e padrões são descritos pelas especificações de eventos e seus relacionamentos. Linguagens de eventos, no entanto, não consideram aspectos espaciais (necessários em biodiversidade) e a composição de eventos é limitada. Tendo em vista esse contexto, a dissertação propõe uma linguagem baseada em lógica para que cientistas especifiquem cenários de interesse. Esses cenários são baseados em composição de eventos complexos. As principais contribuições da dissertação são: proposta da arquitetura de um *framework* para detecção de eventos complexos, que estende o trabalho de Koga [17]; um modelo de dados para representar eventos em biodiversidade; e uma linguagem para descrever padrões de forma hierárquica, explorando o relacionamento espacial e temporal entre os eventos em diferentes níveis de abstração.

Dedico este trabalho a minha irmã, Juliana, por desde pequenina se interessar pela área, ter curiosidade e querer ajudar em meus estudos.

Agradecimentos

Em primeiro lugar, eu gostaria de agradecer à professora Claudia Medeiros pela dedicação e empenho dedicados a este trabalho e, principalmente, pela convivência harmoniosa, apoio, compreensão e ensinamentos concedidos a mim.

Gostaria de agradecer aos colegas do LIS pela criação de um ambiente de trabalho agradável, companheirismo, ajuda e contribuições durante todo o período do mestrado. Em especial a Jaudete, Ive, Patrícia, Matheus, Ivo e Renato pela ajuda, apoio e amizade construída.

Gostaria de agradecer aos membros da banca pela disponibilidade, presença e contribuições. Em especial ao professor André Santanchè pelo ensino e pesquisa desenvolvida no IC-Unicamp e laboratório LIS, contribuindo e acompanhando meu processo de aprendizagem.

Gostaria de agradecer minha família e amigos que incentivaram e acreditaram nesta conquista. Aos meus pais e irmãs, Rose, Wêlds, Monick e Juliana que são meus maiores motivadores. Aos meus tios Leonardo, Iara, André e Iolanda. Agradeço ao meu companheiro Lucas pelo apoio e por poder compartilhar este e muitos outros momentos de nossas vidas. E a todos os meus amigos, em especial Anderson, Lucas Miguel, Isabela, Hanna, David, Aline, Márcio, Luana e Thainná que acompanharam de perto o processo para esta conquista.

Finalmente, gostaria de agradecer às agências de fomento pelo suporte financeiro que, direta ou indiretamente contribuíram para a realização desta pesquisa: FAPESP (2013/02269-7), FAPESP/Cepid em Ciência e Engenharia da Computação (2013/08293-7), Instituto Virtual de Pesquisa FAPESP-Microsoft (projeto NavScales - 11/52070-7), CNPq (projeto MuZOO), FAPESP-PRONEX (projeto eScience), INCT em Ciência Web e CAPES. As opiniões expressas neste trabalho não necessariamente refletem as opiniões das agência de fomento.

Sumário

Abstract	ix
Resumo	xi
Dedication	xiii
Agradecimentos	xv
1 Introdução	1
2 Revisão Bibliográfica	4
2.1 Detecção de Eventos	4
2.2 Processamento de Eventos Complexos	7
2.2.1 Visão Geral	7
2.2.2 Estrutura dos Eventos	8
2.2.3 Descrições de Padrões	10
3 Proposta e Modelo de Linguagem para Detecção de Padrões Espaço-Temporais	14
3.1 Visão Geral do <i>Framework</i> para Detecção de Eventos	14
3.2 Definição de Eventos Simples	16
3.3 Definição de Eventos Complexos	19
3.4 Linguagem para Especificação de Padrões Espaço-Temporais	23
3.4.1 Gramática	24
3.4.2 Semântica dos Elementos da Linguagem	27
4 Estudo de Caso	49
4.1 Cenário 1: Chuva forte	50
4.2 Cenários 2 e 3: Chuva Forte Incluindo Aspectos Espaciais	53
4.3 Cenário 4: Frente Fria em Campinas	56

5	Conclusões e Trabalhos Futuros	60
5.1	Conclusões	60
5.2	Trabalhos Futuros	60
	Referências Bibliográficas	63

Lista de Tabelas

4.1	Eventos no corte <i>Fluxo1</i> do fluxo de eventos	50
4.2	Evento <i>ec1</i> adicionado ao <i>Fluxo1</i> , parte 1	51
4.3	Evento <i>ec1</i> adicionado ao <i>Fluxo1</i> , parte 2	51
4.4	Estrutura do evento <i>ec2</i> , parte 1	51
4.5	Estrutura do evento <i>ec2</i> , parte 2	51
4.6	Eventos associados a <i>Fluxo2</i> , escondidos os atributos irrelevantes para o padrão	54
4.7	Estrutura dos eventos <i>ec3</i> a <i>ec6</i> , parte 1	54
4.8	Estrutura dos eventos <i>ec3</i> a <i>ec6</i> , parte 2	55
4.9	Cenário antes, durante e após a passagem de uma frente fria, adaptado de Ahrens (1994) apud [28]	56
4.10	Eventos associados a <i>Fluxo3</i>	58
4.11	Estrutura dos eventos <i>ec7</i> a <i>ec9</i> , parte 1	58
4.12	Estrutura dos eventos <i>ec7</i> a <i>ec9</i> , parte 2	58

Lista de Figuras

2.1	Arquitetura básica do processamento de eventos, adaptado de [11].	8
2.2	Estrutura de um evento, recorte extraído de [11].	9
2.3	Estrutura de eventos em CEP, extraída de [34].	10
2.4	Relações temporais entre intervalos definidas por Allen [4].	12
3.1	Arquitetura adaptada de [17].	15
3.2	Estrutura de um Evento Simples.	17
3.3	Estrutura de um Evento Complexo.	20
3.4	Cenário 1 para Evento Complexo de Chuva Forte.	22
3.5	Cenário 2 para Evento Complexo de Chuva Forte.	22
3.6	Árvore sintática do padrão <i>ex</i> gerada com a ferramenta ANTLR.	28
3.7	Semântica do operador <i>before</i> entre intervalos e pontos.	32
3.8	Semântica do operador <i>meet</i> entre intervalos.	33
3.9	Semântica do operador <i>overlap</i> entre intervalos.	33
3.10	Semântica do operador <i>finish</i> entre intervalos.	34
3.11	Semântica do operador <i>during</i> entre intervalos.	35
3.12	Semântica do operador <i>start</i> entre intervalos.	35
3.13	Semântica do operador <i>equal</i> entre intervalos e pontos.	36
3.14	Objetos com <i>buffer</i> de 1km.	37
3.15	Operadores espaciais sobre conjuntos de objetos são aplicados sobre os MBR dos conjuntos.	37
3.16	Semântica do operador <i>north</i> entre dois objetos <i>a</i> e <i>b</i>	38
3.17	Semântica do operador <i>east</i> entre dois objetos <i>a</i> e <i>b</i>	38
3.18	Relacionamento <i>disjoint</i> entre objetos topológicos.	40
3.19	Relacionamento <i>disjoint</i> [buf] entre objetos topológicos.	40
3.20	Relacionamento <i>touch</i> entre objetos topológicos.	41
3.21	Relacionamento <i>touch</i> [buf] entre objetos topológicos.	42
3.22	Relacionamento <i>overlap</i> entre objetos topológicos.	43
3.23	Relacionamento <i>overlap</i> [buf] entre objetos topológicos.	43
3.24	Relacionamento <i>inside</i> entre objetos topológicos.	44

3.25	Relacionamento <i>inside</i> [buf] entre objetos topológicos.	44
3.26	Relacionamento <i>cross</i> entre objetos topológicos.	45
3.27	Eventos em uma janela temporal de 3 segundos.	47
3.28	Exemplos de padrões hierárquicos.	48
4.1	Evento Complexo gerado pela verificação do padrão <i>acumChuvaHMaior15</i>	51
4.2	Evento Complexo gerado pela verificação do padrão <i>chuvaforte</i>	51
4.3	Mapa de Campinas e Região.	54
4.4	Eventos complexos gerados pelos padrões <i>chuvaForteViz</i> e <i>chuvaForteLoc</i>	55
4.5	Evento complexo gerado pelo padrão <i>frenteFriaCamp</i>	58

Lista de Siglas

AAL *Ambient Assisted Living*

ABNF *Augmented Backus-Naur Form*

ANTLR *ANother Tool for Language Recognition*

CED *Complex Event Detection*

CEDR *Complex Event Detection and Response*

CEP *Complex Event Processing*

CPU *Central Processing Unit*

EPL *Event Processing Language*

ESB *Enterprise Service Bus*

FPGA *Field-Programmable Gate Array*

MBR *Minumum Bounding Rectangle*

NDVI *Normalized Difference Vegetation Index*

Capítulo 1

Introdução

De modo geral, biodiversidade se refere à abundância, distribuição e interações entre genótipos, espécies, comunidades, ecossistemas e biomas. Assim, aplicações de biodiversidade se caracterizam por necessitar de uma grande variedade de dados ambientais e de espécies – por exemplo, nomes taxonômicos, dados de coleta, sons de animais, dados meteorológicos e outros. Em incontáveis problemas na biodiversidade, esses dados são coletados e analisados em múltiplas escalas no espaço e no tempo, correlacionando variáveis ambientais a seres vivos e seus *habitats* [14]. Embora a maioria destes dados seja gerada por sensores em tempo real, há um sem-número de análises que exigem a correlação com dados de outros tipos, cuja escala temporal de variação é bem maior (às vezes, séculos). Exemplos são dados geológicos, hidrográficos ou de relevo.

O monitoramento ambiental, por exemplo, é uma das aplicações de biodiversidade que requer a análise de dados multiescala, desde imagens de satélites a vários fluxos de dados de estações meteorológicas [25]. Dependendo do objetivo da aplicação, uma variedade de sensores pode ser utilizada. Tipicamente, incluem temperatura, umidade, luz, pressão, vibração, qualidade do ar, entre outros. Quando esse monitoramento envolve animais, há também outros sensores, por exemplo, de movimento ou de som.

Um problema em aberto neste contexto é como especificar e detectar cenários de interesse (como mudanças climáticas, desmatamento, incêndio florestal ou poluição da água) a partir de variáveis ambientais, em múltiplas escalas, para facilitar aos cientistas a análise de fenômenos e correlações com dados coletados em campo. Além disso, uma detecção em tempo real de fenômenos físicos e climáticos permite tomadas de ações que minimizam o impacto no ambiente.

Para ajudar a solucionar o problema, este trabalho usa Processamento de Eventos Complexos (*Complex Event Processing* - CEP) para processar fluxo de dados gerados por sensores meteorológicos. Em CEP, todos os dados são tratados como eventos com o principal objetivo de detectar certos eventos em tempo quase real, sinalizando uma

situação de interesse [34]. Os cenários de interesse são descritos por padrões de eventos (ou simplesmente padrões) que são verificados sobre o fluxo de eventos, buscando por aqueles que satisfazem ao padrão descrito. Essa é uma tarefa desafiadora; havendo um grande conjunto de eventos, localizar tais padrões por inspeção obviamente não é viável [38]. Além disso, em CEP a detecção não se restringe a uma busca de eventos isolados. Mecanismos de Detecção de Eventos Complexos (*Complex Event Detection* - CED) permitem a composição e busca por vários eventos que possuam relações entre si. A partir de dados ambientais, um exemplo de cenário de interesse a ser detectado é a aproximação de uma frente fria que envolve, entre outros aspectos, uma queda de temperatura em uma certa região seguida por chuva forte e mudança da direção do vento. Uma vez que eventos respeitando essa descrição são detectados por um processamento em CEP, o evento complexo referente a essa ocorrência é gerado e sinalizado ao usuário interessado.

Apesar de várias propostas de detecção e especificação de padrões em várias situações, falta uma maior sistematização dos padrões de eventos, que leve em consideração combinações de eventos com variação na escala de espaço e tempo. Assim, o objetivo desta dissertação é contribuir para estudos em biodiversidade, concentrando-se na detecção de eventos em tempo real e especificação de padrões de eventos ambientais, a serem a seguir correlacionados com outros tipos de dados.

O processamento de eventos será realizado em um *framework*, que é uma extensão ao trabalho de Koga [17], que desenvolve uma estrutura para integração de dados de fontes heterogêneas, mas é limitado à detecção de eventos primitivos. Esta dissertação complementa o trabalho anterior adicionando a composição e detecção de eventos complexos, o que permite a detecção de cenários mais elaborados. Para cobrir a maioria das situações de interesse no contexto de biodiversidade, este trabalho especifica uma linguagem baseada em lógica para escrever padrões de forma hierárquica e relacionar eventos entre si, no tempo e no espaço. A literatura de CEP não considera relacionamentos espaciais entre eventos. Além disso, apresenta propostas de propósito geral simples em termos de expressividade de padrões (e.g. [38, 30]) ou propostas mais completas porém limitadas a certo tipo de aplicação, modelando seu contexto, por exemplo, por ontologias em um mundo fechado e bem definido (e.g. [34]).

As principais contribuições desta pesquisa são:

- definição da arquitetura de um *framework* para detecção de eventos complexos;
- definição de um modelo de representação para eventos em aplicações de biodiversidade;
- a especificação de uma linguagem de processamento de eventos para especificar cenários de interesse nesse contexto por meio de padrões hierárquicos, explorando o

relacionamento espacial e temporal entre os eventos em diferentes níveis de abstrações.

O trabalho gerou as seguintes publicações:

- Santo, J. ; Medeiros, C. Complex Pattern Detection and Specification from Multiscale Environmental Variables for Biodiversity Applications. Proc. of CSBC 2014 - BreSci, Sociedade Brasileira de Computação (SBC), Brasília-DF, 2014, ISSN: 2175-2761 (resumo estendido).
- Santo, J. ; Medeiros, C. Complex Pattern Detection and Specification to Support Biodiversity Applications. Proc of SBBD 2014 - WTDBD, Sociedade Brasileira de Computação (SBC), Curitiba-PR, 2014, ISSN: 2316-5170 (resumo estendido).
- Santo, J. ; Medeiros, C. Complex pattern detection and specification from multiscale environmental variables for biodiversity applications. Proc of IEEE e-Science 2014, Guarujá-SP, 2014 (pôster).

O restante da dissertação está organizado como segue. O capítulo 2 apresenta a revisão da literatura em torno do tema de Detecção de Eventos. O capítulo 3 apresenta a contribuição da pesquisa na detecção de eventos, com o detalhamento do *framework* e a especificação da linguagem. O capítulo 4 apresenta o estudo de caso representando os cenários de chuva forte e frente fria pela linguagem proposta. O capítulo 5 apresenta as conclusões e direções futuras da pesquisa.

Capítulo 2

Revisão Bibliográfica

A revisão bibliográfica trata da literatura de detecção de eventos de modo geral (seção 2.1), focando posteriormente na Detecção de Eventos Complexos da literatura de CEP (seção 2.2). A Detecção de Eventos Complexos é chamada também de Detecção de Padrões. A dissertação aborda o tema, independente do termo utilizado na literatura, analisando a estrutura dos eventos e descrição de padrões.

2.1 Detecção de Eventos

Evento é uma ocorrência dentro de um sistema ou um domínio particular, é algo que aconteceu em algum domínio. A detecção de eventos é o processo de identificar a ocorrência de eventos específicos, isso é, detectar cenários de interesse. No entanto, esse processo não é trivial; pesquisas vêm atacando diversos desafios inerentes à detecção de eventos utilizando diferentes métodos e aplicando em diferentes contextos.

Kerman et al. [15] apontam alguns desses desafios: 1) Dependência da situação: parâmetros, variáveis e métricas de saída são selecionadas de acordo com um domínio específico; 2) Criticidade da aplicação: frequentemente as aplicações requerem alta precisão em um tempo curto (tempo-real); 3) Diversidade e grande número de fontes de dados: dados não estruturados, texto, imagem, áudio, vídeo, dados espaço-temporais, etc.; 4) Topologia da rede: necessidade de manutenção da estrutura e conexão entre as estações espalhadas em uma região; 5) Algoritmos de detecção: requer pontualidade, alta taxa de detecção e baixa taxa de alarme falso.

O trabalho desenvolvido nesta dissertação enfrenta principalmente os desafios 1 e 3 ao lidar com detecção de eventos em tempo real no contexto de biodiversidade. O desafio 4 foi previamente abordado no trabalho de Koga [17], o qual estendemos, e os desafios 2 e 5 são considerados nos trabalhos futuros propostos.

Nasridinov et al. [24] classificam os vários métodos usados para detecção de eventos

em três categorias: 1) estatístico, 2) probabilístico, 3) inteligência artificial e aprendizado de máquina. Kerman et al. [15] também consideram uma quarta categoria: 4) composto. No método estatístico, a detecção de eventos ocorre quando o parâmetro monitorado excede um valor limite predeterminado. No método probabilístico, a probabilidade da ocorrência de um evento é considerada no processo de detecção. Os métodos de inteligência artificial e aprendizado de máquina requerem algoritmos computacionalmente mais avançados e dirigidos às informações. O método composto combina técnicas de duas ou mais categorias.

Dentre os variados contextos de aplicações em detecção de eventos estão: processo empresarial, segurança, monitoramento de saúde, de rede e monitoramento ambiental. A detecção de eventos, em tempo real ou não, é um dos principais tópicos abordados na literatura de Inteligência Empresarial (*Business Intelligence*), por exemplo no monitoramento de bolsa de valores. Abordagens mais recentes já consideram os diversos outros contextos. Na segurança, por exemplo, Sen et al. [34] aplicam sua proposta em ambientes assistidos (*Ambient Assisted Living* - AAL) detectando eventos como: chuveiro ligado por muito tempo sem ninguém no banho. Em sua proposta de recomendação de padrões, Sen et al. [34] usam CEP para a detecção de eventos em tempo real e ontologia para definir os eventos do contexto. Na saúde, Zoumboulakis e Roussous [39] aplicam sua proposta na detecção de problemas em eletrocardiogramas, entre outros contextos previstos. Os autores propõem a detecção de eventos complexos usando um método probabilístico e, posteriormente, aplicando métricas de similaridade.

Assim como em Inteligência Empresarial, o monitoramento de rede também constitui numerosas aplicações de detecção de eventos na literatura; exemplos são:

- Sauvageon et al. [31] detectam a exata localização do pico de temperatura em um prato de alumínio. Esse trabalho compara a precisão de métodos das 3 categorias apresentadas acima (exceto o método composto) em uma rede de sensores com nós falhos e ruído;
- Diversas pesquisas detectam vários tipos de eventos (desastres naturais, epidemias, condições de tráfego, entre outros) a partir do fluxo de dados em redes sociais (como o *Twitter*) usando diferentes métodos [26, 23, 1]. Por exemplo, Aggarwal e Subbian [1] usam mineração e agrupamento (considerados pelos autores como métodos da categoria 3) na detecção de eventos em dados de rede sociais, considerando não apenas o conteúdo textual da rede, mas também sua estrutura;
- Aplicações de monitoramento de internet (acesso, falha, ataques, etc.) e monitoramento de tráfego. Por exemplo, Dunkel [9] aplica sua proposta no monitoramento de tráfego combinando dados de tráfego, velocidade dos veículos e clima. O autor

propõe uma arquitetura dirigida a eventos em redes de sensores usando CEP para detecção de eventos em tempo real, e ontologias para a estrutura dos eventos.

Alguns tópicos abordados pela literatura de detecção de eventos no contexto alvo desta dissertação, monitoramento ambiental, são apresentados a seguir:

- Monitoramento da qualidade da água: Eliades et al. [10] detectam contaminação na água bebível em tempo real, monitorando a concentração de cloro na água. Os autores adotam um método composto em sua abordagem, usando aprendizado de máquina no entendimento da dinâmica de queda da concentração de cloro, e usando limiares estatísticos na detecção da contaminação da água. Com o mesmo objetivo, o sistema Canary [36] realiza também previsão da qualidade da água usando uma abordagem probabilística;
- Detecção de salinidade: Kerman et al. [16] detectam o grau de salinidade em dados oceanográficos, aplicando métodos estatísticos em abordagens estática, dinâmica e combinada;
- Detecção e previsão de fenômenos naturais, como tempestade, tornados e furacões: Li et al. [19] usam mineração de dados e processamento de eventos em tempo real, prevendo a ocorrência de tempestades buscando por mesociclones. Llaves e Renschler [21] usam ontologia na definição de eventos gerados por geossensores e CEP na detecção de mudanças em tempo real, aplicando sua proposta na detecção de inundações.

Na literatura apresentada, eventos mais elaborados são detectados usando técnicas sofisticadas de aprendizado de máquina ou apenas eventos mais simples são detectados. Em geral, relacionamentos espaço-temporais entre eventos não são explorados; os dados recebidos são processados como captura independente do mundo real, e restrições espaciais são possíveis apenas pela seleção de sensores específicos, quando se conhece sua localização a partir da estrutura da rede.

Esta dissertação adota a tecnologia CEP para detecção de eventos visando explorar o processamento em tempo real e a construção de cenários mais elaborados, explorando os relacionamentos, inclusive espaciais e temporais, entre eventos. Em CEP, eventos são detectados de acordo com limiares específicos, que podem ser definidos por métodos estatísticos ou por modelos definidos pela aplicação. O restante deste capítulo aprofunda a revisão bibliográfica na detecção de eventos em diversos contextos e aplicações, em especial os focados no uso de CEP.

2.2 Processamento de Eventos Complexos

O processamento de eventos é o conjunto de operações sobre eventos, como leitura, criação, transformação e exclusão de eventos. O Processamento de Eventos Complexos é uma tecnologia de processamento de eventos que visa permitir inferências de eventos significativos em um nível mais alto (eventos complexos), a partir de uma sequência de eventos em níveis mais baixos [37]. Por responder em tempo mais hábil, de acordo com Etzion e Niblett [11], CEP é frequentemente utilizado em situações que envolvem o monitoramento de dados de sensores.

2.2.1 Visão Geral

Em processamento de eventos, a palavra *evento* é usada para a entidade de programação, também chamada de objeto de evento, que representa essa ocorrência [11]. Os eventos são classificados em primitivos (básicos) e complexos (compostos). Eventos primitivos representam uma ocorrência em um determinado espaço e tempo. Eventos complexos são formados por uma combinação de eventos primitivos ou complexos utilizando um conjunto de operadores de composição de eventos [3]. Eventos primitivos representam observações de fora do sistema de eventos, enquanto os eventos compostos representam padrões de eventos definidos dentro do sistema [30]. Operadores e regras para a composição dos eventos dependem da linguagem de padrões, apresentados pela seção 2.2.3.

Segundo Etzion e Niblett [11], aplicações orientadas a eventos seguem uma mesma arquitetura básica, ilustrada pela Figura 2.1. As aplicações contêm componentes que geram eventos, chamados de *agentes produtores de eventos*. Os produtores de eventos podem ser sensores de *hardware*, que produzem eventos quando detectam certas ocorrências físicas; instrumentos de *software*, que produzem eventos em condições de erro ou quando exigido pela lógica de programação; ou mesmo humanos. Os *agentes consumidores de eventos* são componentes que recebem os eventos e realizam ações sobre eles. Consumidores podem armazenar os eventos para uso posterior, exibi-los na interface de usuário ou realizar outras ações predefinidas pela aplicação. Muitas vezes, algum processamento é realizado na transmissão de um evento para o agente consumidor. O processamento é realizado pelo *agente de processamento de eventos* e sua principal tarefa é a Detecção de Eventos Complexos.

CED é a tarefa de filtrar eventos, identificando aqueles que são significativos para o domínio da aplicação. A detecção ocorre através do casamento de eventos com padrões. Os padrões representam modelos de cenário de interesse de uma aplicação ou usuário, compostos pelas especificações dos eventos e de suas relações. Além disso, o casamento de padrões permite a construção de eventos complexos a partir de eventos primitivos [9]. Essa possibilidade de composição e construção de eventos é uma das vantagens do CEP.

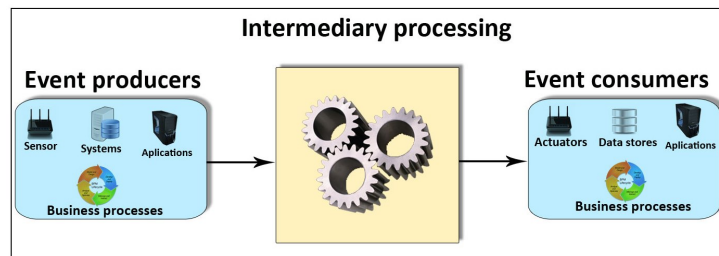


Figura 2.1: Arquitetura básica do processamento de eventos, adaptado de [11].

Decisões sobre o casamento são tomadas de acordo com as *políticas de padrões*, por exemplo, quando mais de um conjunto de eventos satisfaz o padrão. A escolha de uma política depende da aplicação [11]:

- Política de avaliação: determina quando os conjuntos de casamento de eventos são produzidos;
- Política de cardinalidade: determina quantos conjuntos de casamento de eventos são produzidos em uma única partição do contexto;
- Política de tipo repetido: determina o que acontece se a etapa de casamento de eventos encontrar vários eventos do mesmo tipo;
- Política de consumo: determina o que acontece com o evento participante após ele ser incluído no conjunto de casamento;
- Política de ordem: especifica como a ordem temporal é definida.

A apresentação que se segue foca a estrutura dos eventos e a descrição dos padrões em CEP.

2.2.2 Estrutura dos Eventos

Todos os dados podem ser tratados como eventos, independente de contextos e camadas de organização, como registros de vendas, mensagens de texto, informações do mercado de ações, dados meteorológicos, etc. A estrutura dos eventos depende dos parâmetros exigidos para cada contexto da aplicação. Esta seção apresenta estruturas de eventos propostas na literatura de CEP.

Etzion e Niblett [11] definem um conjunto de atributos para a estrutura geral de um evento. Essa estrutura é dividida em três partes: cabeçalho, descritores de atributos e conteúdo aberto. O cabeçalho consiste em metainformação sobre o evento. Os tipos de

dados para representar os valores dos atributos do cabeçalho são: *string*, para o atributo identificador do evento; valores booleanos, para verificador de composição ou não do evento; e *timestamp* e sua granularidade, para registros de tempo. Os atributos do cabeçalho são: tempo de ocorrência, estimativa de certeza do evento, anotação (texto livre), identificador, tempo de detecção (quando o evento é reconhecido pelo sistema de processamento) e gerador de evento. O descritor de atributo depende de cada tipo de evento; ele contém a informação específica sobre o evento, por exemplo, o atributo “temperatura” para armazenar o valor da temperatura medida num dado momento, caso o gerador seja um sensor de temperatura. Por fim, o conteúdo aberto é uma seção opcional para informações adicionais em formato livre. A Figura 2.2 ilustra essa estrutura.

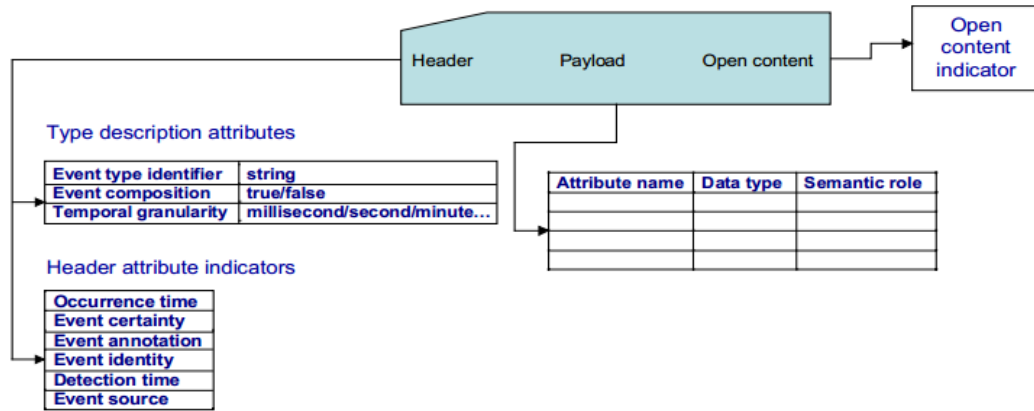


Figura 2.2: Estrutura de um evento, recorte extraído de [11].

Koga [17] especifica a estrutura de eventos básicos no contexto de aplicações ambientais a partir de quatro atributos: valor medido, natureza da variável, variável de espaço e variável de tempo. Enquanto os estudos anteriores consideram eventos instantâneos, Barga e Caituiro-Monge [5], Li et al. [18] e Sen e Stojanovic [34, 32] utilizam uma abordagem baseada em intervalos para representar a ocorrência de eventos. A Figura 2.3, extraída de [34], mostra a ontologia que representa a estrutura de eventos definidos pela 7-tupla $E := (I, N, ST, ET, ES, EO, T)$, onde:

- I é um identificador numérico do evento;
- N é um atributo alfanumérico para o nome do evento;
- ST é um atributo numérico para o tempo de início do evento;
- ET é um atributo numérico para o tempo final do evento;
- EO é uma referência ao operador;

- ES é uma referência para o gerador do evento;
- T é uma referência ao tipo de evento, representados por uma lista de atributos específicos para caracterizar uma classe de eventos.

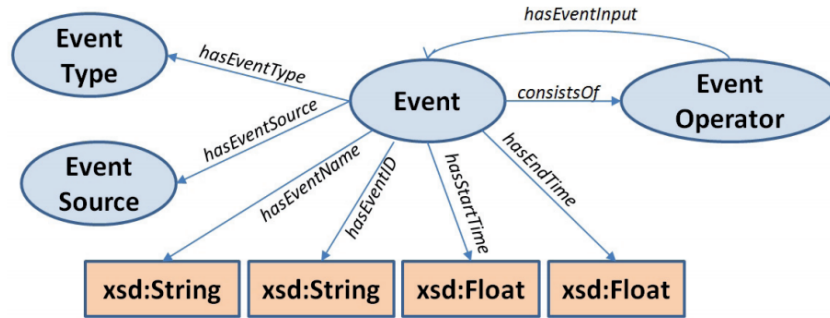


Figura 2.3: Estrutura de eventos em CEP, extraída de [34].

Diferente dos demais trabalhos citados, Barga e Caituiro-Monge [5] não armazenam o tempo da ocorrência do evento, mas sim sua validade. Dessa forma, os eventos têm os campos *Start_Valid_Time* e *End_Valid_Time*, representando o início e fim do intervalo em que o evento pode ser considerado pelo sistema. Usando esse esquema temporal, os autores propõem uma linguagem, denominada CEDR (*Complex Event Detection and Response*), para expressar consultas que filtrem, gerem e correlacionem eventos complexos para aplicações de monitoramento em geral. Essa ideia de período de validade foi retomada por nosso modelo.

2.2.3 Descrições de Padrões

A composição de um padrão de eventos complexos é descrita e estruturada pela Linguagem de Processamento de Eventos (*Event Processing Language* - EPL). Nas linguagens, os relacionamentos entre os eventos são descritos pelos operadores de composição. Existem diversas pesquisas que se empenham em definições de linguagens mais rápidas e expressivas. As soluções propostas têm particularidades na semântica dos operadores e na forma de composição dos padrões, ou seja, diferentes operadores são adotados e eles podem ser tratados de maneira distinta pela linguagem. Esta seção apresenta um levantamento dos operadores e/ou estruturas utilizadas na literatura.

Em abordagens baseadas em autômatos, são adotados os operadores de expressões regulares. Pesquisas usando essa abordagem adicionam mais operadores à linguagem ou definem semântica diferente para as expressões regulares, visando maior expressividade e simplicidade na composição de padrões. Por exemplo, Pietzuch et al. [30] adiciona

o operador de temporização $(E1, E2)_{T1=timespec}$ aos operadores de expressões regulares para permitir a detecção de combinação de evento em uma janela de tempo. Por ter um conjunto limitado e bem definido de operadores, essa abordagem é também utilizada em pesquisa com foco em otimização. Por exemplo, Woods et al. [37] implementam a detecção de eventos complexos em Arranjos de Portas Programável em Campo (*Field-Programmable Gate Arrays* - FPGAs) para dados de tráfego na internet.

A abordagem baseada na lógica ou em regras permite linguagens mais ricas em relação às que são baseadas em autômatos. Obweiger et al. [27], por exemplo, definem a composição de eventos complexos usando uma linguagem baseada em regras. Seu modelo permite aos usuários compor padrões usando uma interface que abstrai as definições dos sub-padrões. A abordagem permite a criação de padrões mais elaborados e facilita o reuso de padrões em diferentes contextos e a criação de bibliotecas de padrões. O trabalho realizou um estudo de caso no domínio de detecção de fraudes.

Baseados na lógica, os padrões são descritos por “combinações” de predicados sobre eventos. Um evento corresponde a um padrão se os valores das suas variáveis satisfazem ao predicado. Assim, a composição de um padrão precisa considerar: (a) os conectores, (b) os quantificadores e (c) os operadores elementos de um evento. Como se verá no Capítulo 3, a linguagem proposta na dissertação segue a mesma linha.

Conectores e Quantificadores

Padrões simples podem ser definidos por um único predicado que busca um evento básico. Padrões compostos podem ser definidos por combinação de predicados através de conectores. Os conectores são operadores lógicos usados na combinação de duas ou mais sentenças de predicados. Na literatura, os principais conectores são \wedge (conjunção), \vee (disjunção) e \neg (negação). Os principais quantificadores encontrados na literatura são os mesmos da lógica:

- \forall (quantificador universal): todos os eventos devem satisfazer ao predicado quantificado por \forall ;
- \exists (quantificador existencial): pelo menos um evento deve satisfazer ao predicado quantificado por \exists .

Outros quantificadores são definidos com diferentes semânticas. Por exemplo, em Etzion e Niblett [11], a semântica dos quantificadores *all*, *any*, *absence* é definida para quantificar uma lista de predicados:

- *all*: o padrão é detectado quando existe pelo menos um evento que satisfaça cada predicado da lista de predicados;

- *any*: o padrão é detectado quando existe pelo menos um evento que satisfaça pelo menos um dos predicados da lista;
- *absence*: padrão é detectado quando nenhum dos eventos satisfaz a qualquer predicado da lista.

Barga e Caituiro-Monge [5] definem adicionalmente outros dois quantificadores: *atleast* e *atmost*. Para esses quantificadores, um valor natural n é definido, e o padrão é satisfeito desde que existam no mínimo e no máximo, respectivamente, n eventos que satisfaçam o predicado.

Operadores Temporais

A maioria dos trabalhos em CEP utilizam as relações temporais de Allen [4] como operadores para a composição de padrões temporais variados. Allen define treze operadores que expressam as relações entre dois intervalos de tempo. Os operadores, como mostrado na figura 2.4, são: *before*, *after*, *meets*, *met-by*, *overlaps*, *is-overlapped-by*, *finishes*, *is-finished-by*, *contains*, *during*, *starts*, *is-started-by* e *equals*.



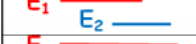
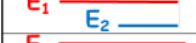
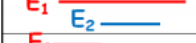


	E_1 before E_2	E_2 after E_1
	E_1 meets E_2	E_2 is-met-by E_1
	E_1 overlaps E_2	E_2 is-overlapped-by E_1
	E_1 is-finished-by E_2	E_2 finishes E_1
	E_1 contains E_2	E_2 during E_1
	E_1 starts E_2	E_2 is-started-by E_1
	E_1 equals E_2	E_2 equals E_1

Figura 2.4: Relações temporais entre intervalos definidas por Allen [4].

Outros trabalhos adotam apenas um operador de sequência, como em Barga e Caituiro-Monge [5], que equivale à semântica de precedência do operador *before* aplicadas sequencialmente. Ainda outros trabalhos estendem as relações de Allen com outros operadores. Por exemplo, Li et al. [18] introduzem o operador ISEQ como uma forma agregada de aplicar as relações de Allen em sequência. ISEQ pode combinar, por exemplo, $A \text{ meet } B$ e $B \text{ overlap } C$, e assim por diante. Os autores apresentam a implementação desse operador visando melhorar o desempenho da memória e CPU (*Central Processing Unit*).

Padrões temporais podem representar cenários como queda de temperatura, onde o valor medido diminui no decorrer do tempo. Isto é, para todos os eventos Ei , $Ei.v > Ej.v \mid j = i + 1$.

Operadores Espaciais

Embora padrões espaciais sejam conceitualmente abordados por Etzion e Niblett [11], não são implementados na literatura de CEP. Os operadores abordados em Etzion e Niblett [11] são as relações topológicas e métricas bem consolidadas na literatura de banco de dados espaciais ([13, 7, 8]) e espaço-temporais ([12, 29]). Guting [13] classifica as relações espaciais em:

- Topológica, como *adjacente*, *dentro* e *disjunto*;
- Direcionais, como *acima*, *abaixo*, *a_norte_de* e *a_leste_de*;
- Métricas, como “distância<100”

Esta dissertação considera um conjunto de padrões espaciais para verificar a ocorrência de algum evento em locais específicos, como aqueles que distam x *km* de outro local. Padrões espaço-temporais podem verificar a existência de um movimento, como deslocamento ao longo do tempo de uma frente fria.

Capítulo 3

Proposta e Modelo de Linguagem para Detecção de Padrões Espaço-Temporais

No contexto de biodiversidade, um mecanismo para especificar e detectar cenários de interesse deve suportar o processamento de dados de fontes heterogêneas e a construção de cenários mais elaborados que explorem relações entre as variáveis ambientais no tempo e espaço. Como evidenciado no capítulo anterior, tratamos esse problema no domínio de CEP, onde os dados são tratados como eventos e os cenários como padrões de eventos. Esta dissertação estende o trabalho desenvolvido por Koga [17], que permite a integração de dados de fontes heterogêneas, mas é limitado a padrões de eventos primitivos.

O restante deste capítulo apresenta o modelo desenvolvido para a especificação e detecção de cenários no domínio de monitoramento ambiental para biodiversidade, sendo separado pelas seguintes contribuições, sempre no domínio de monitoramento ambiental para biodiversidade: a) visão geral do mecanismo para detectar cenários de interesse em tempo real (seção 3.1); b) representação dos dados em forma de eventos (seção 3.2); c) representação de cenários através de eventos complexos (seção 3.3); e d) especificação de uma linguagem de detecção de eventos para descrever cenários (seção 3.4);

3.1 Visão Geral do *Framework* para Detecção de Eventos

O *framework* proposto mantém a mesma arquitetura desenvolvida por Koga [17], acrescentando a ela a detecção de eventos complexos. A figura 3.1 ilustra a arquitetura do *framework* estendido, horizontalmente desenhada. A arquitetura de Koga [17] é centrada

em dois aspectos principais: o uso de *Enterprise Service Bus* (ESB) para processar um fluxo de dados de forma uniforme; e a adoção de CEP para detecção de eventos. Dados ambientais são pré-processados e transformados em eventos, que trafegam pelo ESB e são processados por CEP. Estendemos essa proposta provendo alimentação contínua de eventos no barramento, para permitir a composição e detecção de eventos complexos de forma hierárquica.

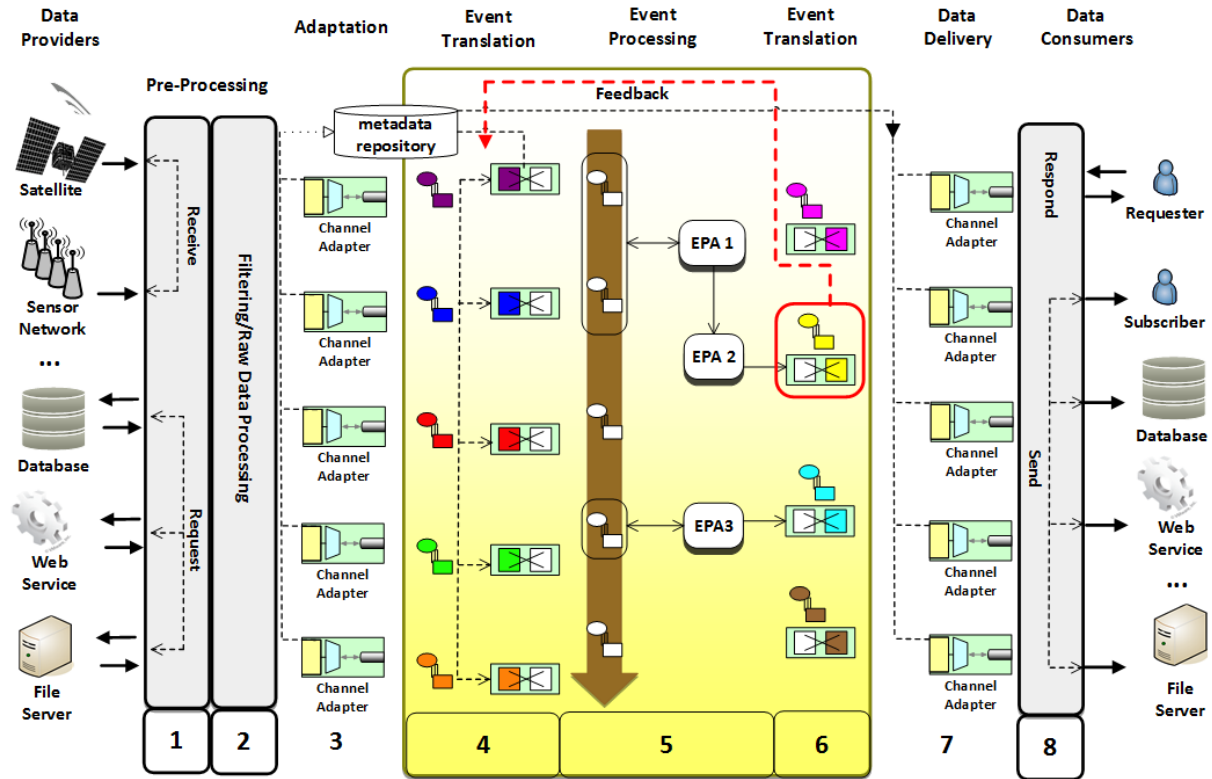


Figura 3.1: Arquitetura adaptada de [17].

Um evento simples, para Koga [17], é transformado em uma mensagem padronizada com as seguintes informações: valor da variável medida, natureza da variável, uma única coordenada geográfica referente ao local de ocorrência do evento e intervalo de tempo de ocorrência do evento. Esse modelo de eventos foca na padronização de eventos primitivos, sendo assim limitado no que se refere aos aspectos de gerenciamento de dados em múltiplas escalas e composição de eventos.

Da esquerda para a direita na Figura 3.1, as etapas 1 e 2 realizam o pré-processamento. Na etapa 1, os provedores de dados são conectados à plataforma. Na etapa 2, os dados são filtrados de acordo com o interesse da aplicação. Na etapa 3, os dados selecionados pela aplicação são encapsulados em mensagens padronizadas pelos adaptadores de canal

(*channel adapter*, esses canais fazem parte do padrão ESB). As etapas de 4 a 5 correspondem à tradução das mensagens em eventos e seu processamento por CEP. Se o padrão é detectado, a etapa 6 encapsula o evento correspondente em uma nova mensagem, seguindo o mesmo padrão. Nas etapas 7 e 8, essa mensagem é padronizada e enviada para o usuário interessado.

Nosso trabalho complementa a arquitetura adicionando a composição e detecção de eventos complexos, ilustrada pela seta *feedback* pontilhada em negrito na Figura 3.1, que retorna o evento detectado ao fluxo de eventos. Ou seja, trata-se de um *feedback* da etapa 6 à etapa 4. Essa extensão do trabalho de Koga [17] provê a descrição de padrões mais representativos, compostos por hierarquias de eventos primitivos e complexos. Quando a composição de eventos é detectada, ela é enviada de volta ao barramento do ESB. Portanto, um evento complexo pode fazer parte de composições mais complexas, gerando eventos compostos em nível mais alto. Na arquitetura original, apenas composições de eventos primitivos poderiam ser detectadas e retornadas ao usuário.

Essa extensão requer esforços consideráveis em *design* e implementação. Um desses esforços, e a principal contribuição desta pesquisa, é a formalização da especificação de eventos e padrões no contexto de biodiversidade. Essa formalização é apresentada no restante deste capítulo (seções 3.2 a 3.4) e é inspirada em propostas aplicadas a diferentes domínios (e.g., [11, 5, 34]). Essa especificação deve: permitir a composição hierárquica de padrões e eventos, como em [34, 27, 20, 38]; combinar dados de fontes heterogêneas, como em [17]; na composição, considerar o local onde o evento ocorre, como em [17]; e explorar as relações temporais e espaciais entre eventos, as últimas até agora não exploradas na literatura de CEP. Além disso, a especificação estende a semântica dos operadores para permitir capturar eventos em múltiplas escalas temporais e espaciais. Dessa forma, essa especificação pode expressar cenários de biodiversidade com complexidade variada, desde situações relativamente simples (valor de um atributo) a situações combinando dados hidrográficos, de vegetação e de relevo.

O Capítulo 4 apresenta um estudo de caso como validação conceitual deste trabalho. A implementação desse mecanismo faz parte dos trabalhos futuros desta pesquisa.

3.2 Definição de Eventos Simples

Como destacado no Capítulo 2, os eventos são classificados em simples e complexos. Um evento simples é a representação de uma ocorrência no mundo real em uma determinada localização e período. Na pesquisa desta dissertação, um evento simples em biodiversidade é a medida de uma variável ambiental em um ponto espaço-temporal. A estrutura proposta para eventos simples é inspirada em vários trabalhos correlatos, sendo a seguinte (Figura 3.2):

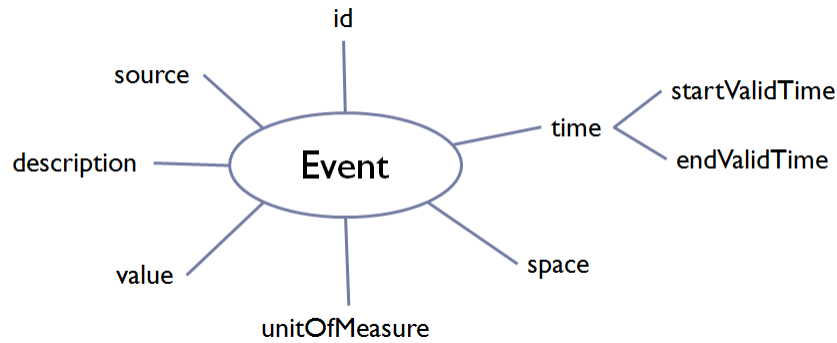


Figura 3.2: Estrutura de um Evento Simples.

- *id*: atributo identificador do evento;
- *source*: produtor do evento.
Ex: banco de dados, sistemas, sensores, satélites, etc.
Essa informação permite determinar a proveniência do evento e possibilita a filtragem de conjuntos de dados de interesse para o usuário. Os diversos produtores de eventos diferem quanto à variável mensurada e à qualidade do dado. Dessa forma, um especialista pode selecionar apenas dados de determinados produtores, uma vez que conhece quais dispositivos estão melhor localizados ou proveem dados mais confiáveis;
- *description*: descrição que informa qual é a variável medida.
Essa informação define qual aspecto do mundo real foi observado e mensurado em um determinado evento.
Ex: Temperatura, velocidade do vento, frequência de som, pH da água, índice de qualidade do solo, etc.;
- *value*: valor medido (numérico ou literal) da variável em *description*;
- *unitOfMeasure*: unidade de medida do valor em *value*.
Esse atributo é necessário pois eventos de naturezas iguais (com mesmo *description* ou equivalente) podem ser armazenados com diferentes unidades de medida, por exemplo, devido a hábitos culturais dos usuários;
- *space*: localização em que o evento acontece.
Esse atributo permite lidar com dados em múltiplas escalas espaciais, sendo representado por um conjunto de coordenadas com latitude e longitude $\{(x_1, y_1), (x_n, y_n)\}$. Esse conjunto indica o local de ocorrência de um evento simbolizado no espaço por

um único ponto (uma coordenada), linha (duas coordenadas) ou polígono (mais de 2 coordenadas);

- *time*: intervalo de tempo em que o evento é considerado válido para o sistema.

Atributo composto por *startValidTime* e *endValidTime*.

startValidTime: *timestamp* de quando o evento acontece ou se inicia.

endValidTime: *timestamp* do fim do intervalo de tempo em que o sistema considera o evento representativo no mundo real. Esse valor depende da natureza do evento, principalmente da sua importância, velocidade ou frequência de mudança no mundo real. Variáveis menos estáveis têm intervalo de validade menor, enquanto variáveis mais estáveis têm intervalo de validade maior.

O significado desse atributo foi adaptado da teoria de CEP. Originalmente, CEP utiliza um intervalo para representar o tempo de duração da ocorrência do evento. Nossa adaptação provê uma maneira para gerenciar eventos gerados em diferentes escalas no tempo.

Os atributos *space* e *unitOfMeasure* não existem no modelo CEP e foram adicionados para permitir modelar eventos de interesse do domínio alvo. O atributo *time* teve sua semântica alterada para o mesmo propósito.

Os eventos descritos por essa representação podem ser resultados diretos de medidas, ou criados a partir de pré-processamento de dados. Por exemplo, imagens de satélite ou sons de animais podem ser representados por descritores. Esses descritores são formados por um conjunto de variáveis e valores extraídos da representação original, o qual pode se tornar um conjunto de eventos simples contendo um único valor para uma única variável no tempo e espaço. Um exemplo desse tipo de evento é o valor médio de um pixel em um polígono de uma imagem de satélite que retrata a biomassa de uma região. O evento correspondente pode, por exemplo, referenciar o valor dessa biomassa.

Exemplos de eventos simples:

- Temperatura: (event1, sensorTemp1, “temperatura externa”, 25.6, “°C”, (21° 41’ 07” S, 46° 15’ 14” W), 06/10/2006 14:05-14:06), retratando que a temperatura externa naquele ponto é de 25.6 °C, com validade de 1 minuto.
- Evento extraído de uma série temporal: um som pode ser representado por descritores, em que um parâmetro pode indicar a frequência máxima do espectrograma derivado: (event2, sensorAudio1, “frequência máxima do áudio do Sabiá”, 120, “Hz”, $\{(x_1, y_1)\}$, 06/10/2006 14:05:32-14:05:32), evento calculado a partir de um espectrograma do canto de um sabiá.

- Evento extraído de uma imagem de satélite: (event3, satellite1, “NDVI”, 0.3, none, $\{(x_1, y_1)(x_2, y_2)(x_3, y_3)(x_4, y_4)\}$, 22/04/2000 00:00 - 08/05/2000 00:00), informando que o índice de vegetação NDVI (*Normalized Difference Vegetation Index*) médio do pixel em um polígono, com validade de 16 dias, é 0.3. Como esse índice é resultado de um cálculo de várias variáveis, não tem unidade de medida, apenas um valor.

Ressalta-se que, nesses exemplos, atributos como *source*, *description* e *unitOfMeasure* estão representados por *strings*. Nada impede, no entanto, que tais informações referenciem termos de ontologias, o que permite processamento computacional dentro de um contexto específico.

3.3 Definição de Eventos Complexos

No contexto de biodiversidade, eventos complexos representam um acontecimento descrito por várias variáveis ambientais. Dado um padrão, eventos complexos são gerados relacionando, entre si, os eventos que satisfazem ao padrão. Os relacionamentos entre eventos são expressos por um conjunto de operadores definidos na linguagem de definição de padrão [3].

Em nossa solução, eventos complexos são definidos por composições hierárquicas de eventos menos complexos, seus “filhos”, relacionados por operadores. Essa abordagem permite que eventos de níveis menores sejam rastreados de volta a partir do evento complexo. Em CEP, essa tarefa de rastreo é chamada *drill down* [22]. Em nosso modelo, os atributos *eventList* e *operatorList* de cada evento mantêm informação apenas dos eventos imediatamente inferiores e de como se combinam, parando o *drill down* nos eventos primitivos.

Dado que eventos complexos não representam uma única observação, sua estrutura é modificada, não aplicando valores para alguns atributos e adicionando elementos que referenciam os eventos filhos. A Figura 3.3 ilustra a proposta da dissertação para eventos complexos de forma hierárquica. Ela estende a proposta de CEP nos atributos *operatorList*, *time* e *space*.

Atributos:

- *id*: mesmo conceito do evento simples;
- *source*: mesmo conceito do evento simples.

Uma vez que eventos complexos são gerados dentro do sistema, em geral, o *source* é o próprio sistema de detecção que construiu o evento complexo a partir de eventos

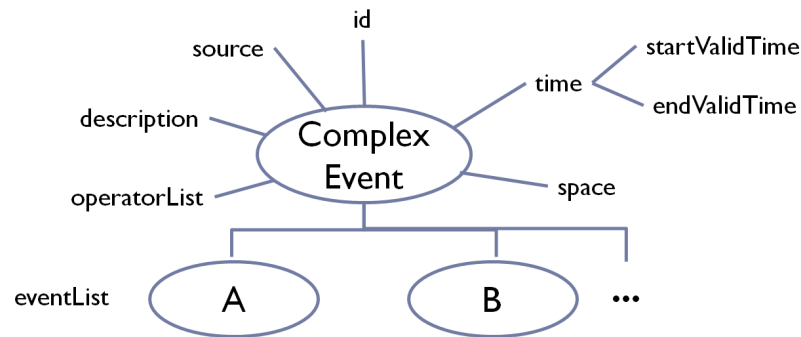


Figura 3.3: Estrutura de um Evento Complexo.

mais primitivos. Algumas vezes pode ser mais específico, como a *thread* ou estação que o processou, no caso de uma aplicação executada em paralelo ou distribuída;

- *description*: mesmo conceito do evento simples.
O nome do padrão é transferido para *description* do evento de nível mais alto. Eventos complexos intermediários gerados pelo casamento do padrão não têm descrição (*description = null*);
- *value* e *unitOfMeasure*: não aplicável.
Esses atributos podem ser diferentes para cada evento que compõe o evento complexo. Os atributos podem ser acessados ao navegar para os eventos inferiores;
- *space*: mesmo conceito do evento simples.
O atributo recebe um valor agregado. O valor é o conjunto de coordenadas dos objetos espaciais resultantes da união ou intersecção dos objetos considerados na relação. A intersecção é aplicada para eventos complexos gerados pelos operadores espaciais *inside* ou *overlap* e a união aplicada para os demais casos;
- *time*: mesmo conceito do evento simples.
Os atributos recebem um valor agregado. O valor é um único intervalo que representa a intersecção ou união de todos os intervalos dos eventos componentes. O intervalo de intersecção é aplicado para eventos complexos gerados pelos operadores temporais *during*, *overlap* ou *equal*, e o intervalo de união é aplicado para os demais casos;
- *eventList*: lista de referências aos eventos “filho” que compõem o evento complexo, ou seja, no nível imediatamente inferior;

- *operatorList*: lista de triplas ordenadas de tipo (a) <atributo, operador, constante> ou (b) <atributo, operador, atributo>, indicando o operador utilizado para relacionar todos os eventos em *eventList*. Os atributos suportados são todos os atributos dos eventos simples: *time*, *startValidTime*, *endValidTime*, *space*, *unitOfMeasure*, *value*, *description*, *source* e *id*. Sobre esses atributos são aplicados operadores literais, numerais, temporais ou espaciais.

Todos os eventos em *eventList* devem obedecer às relações apresentadas na tripla. Um exemplo da primeira construção (a) em uma tripla é <*startValidTime*, *after*, 2015/02/02>, indicando que o evento complexo é formado por eventos (em *eventList*) com tempo de validade inicial após a data (constante) especificada. A segunda construção (b) indica um evento complexo explorando as relações entre os eventos de nível inferior. Por exemplo, a tripla <*startValidTime*, *after*, *endValidTime*>, indica que *A.startValidTime after B.endValidTime* e *B.startValidTime after C.endValidTime*, para sub-eventos *A*, *B* e *C*. Na literatura correlata, as listas de operadores são mais simples, em geral considerando apenas conectores como AND e OR.

Exemplos de Eventos Complexos vão desde ocorrências mais simples, como “chuva forte”(ver a seguir), a ocorrências que envolvem relações espaciais e temporais sobre diversas variáveis ambientais, como movimento de uma frente fria (Capítulo 4).

Exemplo de chuva forte:

Como visto, a formação de eventos complexos é fortemente dependente da descrição do cenário de interesse (padrão). Llaves e Renschler [21] afirmam que a descrição do cenário pode variar de acordo com o pesquisador. Eles exemplificam o fato pela diferença na definição de chuva entre os centros meteorológicos. A *American Meteorological Society*¹ acredita que a intensidade de chuva de pelo menos 0.76cm/h é suficiente para categorizar a chuva como forte. Essa é uma definição mais simples que pode ser descrita por um padrão que busca apenas por um evento primitivo. No entanto, a *Central Weather Bureau de Taiwan* [6] define chuva forte a partir de duas variáveis: chuva acumulada em 1 hora e chuva diariamente acumulada. Essa definição indica que deve ter havido chuva acumulada excedendo 50mm em 24h (evento primitivo), e, dentro dessas 24 horas, ter havido acúmulo de pelo menos 15mm em uma hora (outro evento primitivo).

Tomando a segunda definição, dado que esse evento ocorreu e considerando o cenário mais simples, onde há dados das duas variáveis necessárias: acúmulo por hora e acúmulo por dia, um evento complexo que possivelmente representa esse cenário é ilustrado na Figura 3.4.

¹<http://glossary.ametsoc.org/wiki/Rain>

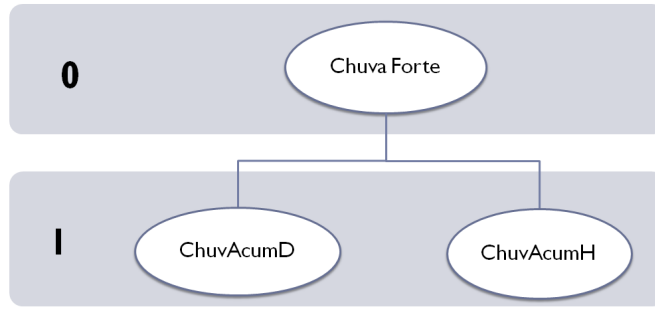


Figura 3.4: Cenário 1 para Evento Complexo de Chuva Forte.

Nesse cenário, os eventos da camada 1 são eventos simples que satisfazem a descrição informal do padrão de Chuva Forte pelo *Central Weather Bureau* [6]. O primeiro evento (*ChuvAcumD*) representa o acúmulo de água, durante um dia, maior que $50mm$. O segundo evento (*ChuvAcumH*) representa o acúmulo de chuva maior ou igual a $15mm$ durante 1 hora. Na camada 0, esses dois eventos são compostos em um único: Chuva Forte. Sua composição envolve um relacionamento temporal assegurando que o evento de duração menor tenha ocorrido dentro da duração do evento de maior intervalo.

Um cenário mais complexo pode ser verificado na Figura 3.5, onde não há informação pré-calculada sobre o acúmulo de chuva por hora. No entanto, os eventos simples *ChuvaInt1*, *ChuvaInt2*, etc. medem o valor de chuva válido por apenas dois minutos. Uma possível maneira de representar o cenário com essas variáveis é simular o cálculo de acúmulo de chuva pela ocorrência de chuva com intensidade maior ou igual a $15mm/h$ durante uma hora. Dessa forma, os eventos de intensidade de chuva são agregados para inferir a informação de acúmulo de chuva por hora.

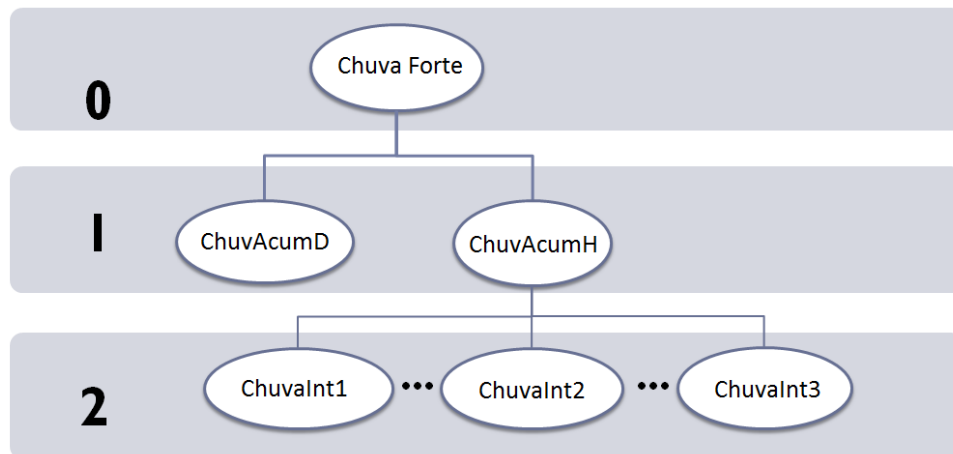


Figura 3.5: Cenário 2 para Evento Complexo de Chuva Forte.

Na Figura 3.5, a camada 2 apresenta apenas 3 (dentre n) eventos simples que ocorreram numa janela de tempo de 1 hora. Para que o evento de chuva forte tenha ocorrido segundo o *Central Weather Bureau* [6], deve haver 30 eventos sucessivos nessa janela de tempo, com intensidade de chuva maior ou igual a $15mm/h$. Esses eventos evidenciam uma hora de chuva com intensidade suficiente para resultar em um acúmulo maior que $15mm$. Na camada 1, esses eventos foram transformados no evento complexo *ChuvInt*. Operadores temporais podem ser também utilizados nessa composição para representar uma sequência na ocorrência dos eventos.

O segundo requisito para haver chuva forte é o acúmulo de água durante um dia ser maior que $50mm$, evidenciado pelo evento *ChuvAcum* na camada 1, como no primeiro cenário. Novamente, a camada 0 compõe o evento Chuva Forte, que pode utilizar operadores temporais para assegurar que todos os eventos requeridos tenham acontecido simultaneamente.

Nesses exemplos, considera-se que todas as observações são de uma única localização. O capítulo 4 apresenta exemplos mais detalhados sobre a construção de eventos complexos. A próxima seção apresenta uma linguagem para especificação de padrões, neste trabalho, utilizada na descrição de cenários de interesse em biodiversidade. A seção apresenta, também, mais detalhes sobre a utilização de operadores espaciais e temporais aplicadas para relacionar eventos e, conseqüentemente, gerar eventos complexos.

3.4 Linguagem para Especificação de Padrões Espaço-Temporais

Uma Linguagem de Processamento de Eventos define as regras de composição de um padrão. O padrão é continuamente verificado em um fluxo de eventos, buscando por um subconjunto que case com as condições descritas pelo padrão. Em seguida, os eventos de cada subconjunto selecionado são combinados para gerar novos eventos complexos. Nesta dissertação, os padrões descrevem cenários de interesse em biodiversidade e os eventos complexos evidenciam a ocorrência desse padrão.

A principal contribuição da linguagem proposta é a melhoria da expressividade e da simplicidade na escrita de cenários, que envolve: (A) explicitar os relacionamentos entre os dados ambientais através de uma escrita compacta de operadores; (B) aumentar a expressividade do padrão através da adição de relacionamentos espaciais entre os eventos; (C) permitir a composição hierárquica de padrões visando uma representação mais clara e o reuso de subpadrões e eventos complexos.

A linguagem visa descrever regras para a especificação de eventos de interesse e suas relações. Portanto, não define regras para, por exemplo, manipulação de fluxo de dados

ou controle de saída dos padrões, que geralmente são incluídas em uma EPL completa. A manipulação de fluxo de eventos requer operações como união de diferentes fluxos ou seleção de tipos de eventos para determinado fluxo. Neste trabalho, a arquitetura já possui uma etapa de pré-processamento, sendo mantida da arquitetura de Koga [17], em que os dados podem ser filtrados em um fluxo de acordo com a fonte ou tipo de dado desejado pela aplicação. Dessa forma, a linguagem proposta apenas permite que um padrão seja executado sobre um único fluxo de eventos, que pode conter dados de uma ou várias naturezas. Por sua vez, o controle de saída envolve regras para decidir qual conjunto de dados de um fluxo processar (o primeiro, último, algum, ou todos encontrados) de forma a buscar o padrão desejado.

Esta seção propõe a especificação de uma linguagem para definição de padrões no processo de detecção de cenários de interesse. A subseção 3.4.1 apresenta a sintaxe para construção de um padrão e a subseção 3.4.2 apresenta a semântica de cada elemento da linguagem no contexto da detecção de padrão.

3.4.1 Gramática

A estrutura da linguagem proposta se baseia na lógica de primeira ordem, em que padrões são descritos por especificações de eventos compostas por combinações de predicados sobre seus atributos. Um evento casa com um padrão se os valores dos seus atributos satisfazem ao predicado. A composição de um padrão precisa considerar os seguintes elementos: conectores, quantificadores e operadores comparativos sobre os atributos de um evento.

A linguagem utiliza quantificadores para limitar a quantidade de eventos necessários para casar com o padrão; conectores para combinar predicados e especificações de eventos; e operadores para construir predicados determinando condições que devem ser satisfeitas, incluindo as relações espaciais e temporais entre eventos. A seguir é apresentada a definição formal da linguagem para composição de padrões utilizando a forma Aumentada do Formalismo de Backus-Naur (ABNF)². Nessa definição, o elemento que representa um período de tempo (<esper_time_period> na gramática abaixo) foi extraído da EPL Esper. Esta gramática foi validada sintaticamente usando a ferramenta ANTLR³. O arquivo utilizado na validação pela ferramenta ANTLR está disponível em <http://www.lis.ic.unicamp.br/~jacqueline/dissertacao/>. Neste *link* encontra-se também as árvores sintáticas geradas a partir dos padrões descritos no estudo de caso (Capítulo 4).

²<https://tools.ietf.org/html/rfc5234>

³Gerador de parser disponível em <http://www.antlr.org/>

```

<pattern> ::= <pattern_name> '=' <set_esp_events> [<time_window>]

<set_esp_events> ::= ¬ '('(<set_esp_events>')'
                  | ¬ <set_esp_events>
                  | (<esp_event> | <pattern_name>) *(<biconnector>
                  ('('<set_esp_events>')'|<set_esp_events>))

<esp_event> ::= <quant> <var_name> ['('(<predicate>')')] '|' <predicate>
              | <quant> <var_name>

<predicate> ::= <condition> *(<biconnector>
              ('('<predicate>')'|<predicate>))
              | ¬ '('<predicate>')'

<condition> ::= <attribute> <comp_operator> <attribute>
              | <comp_operator> '('<attribute> +(','<attribute>) ')'
              | <attribute> <comp_operator> <constant>
              | '('<attribute> +(','<attribute>) ') '<comp_operator>
              (<constant>|<attribute>)

<attribute> ::= <var_name> '.' ('id' | 'source' | 'description'
                              | 'unitOfMeasure' | 'time' | 'startValidTime'
                              | 'endValidTime')
              | <spatial_at> | <value_at>
              | <var_name>
              | <value_at> <arithmetic_op> <constant>
              | <spatial_func>

<value_at> ::= <var_name> '.space'

<spatial_at> ::= <var_name> '.space'

<arithmetic_op> ::= '+' | '-' | '×' | '÷'

<constant> ::= '"'<string>'"' | <number> | <space_object> | <date_time>

<comp_operator> ::= <stnumber_op> | <spatial_op>
                  | <temporal_op>

<stnumber_op> ::= '=' | '≠' | '>' | '<' | '≥' | '≤'

<temporal_op> ::= ('before' | 'meet' | 'overlap' | 'finish' | 'during'

```

```

    | 'start' | 'equal') ['['<esper_time_period>']']

<spatial_op> ::= ('disjoint' | 'touch' | 'overlap' | 'inside' | 'cross'
    | 'north' | 'east') ['['<buf>']']
<buf> ::= <number><length_unit>

<spatial_func> ::= area '(' (<spatial_at>|<space_object>|<var_name>)) ')'
    | perimeter '(' (<spatial_at>|<space_object>|
        <var_name>) ')'
    | length '(' (<spatial_at>|<space_object>|<var_name>)
        ') '
    | distance '(' (<spatial_at>|<space_object>|<var_name>)
        ,(<spatial_at>|<space_object>|<var_name>) ') '

<time_window> ::= '[' <esper_time_period> ']'

<esper_time_period> ::= [<year_part>] [<month_part>] [<week_part>]
    [<day_part>] [<hour_part>] [<minute_part>]
    [<second_part>] [<millisecond_part>]

<year_part> ::= (<int_number> | <var_name>) ('years' | 'year')
<month_part> ::= (<int_number> | <var_name>) ('months' | 'month')
<week_part> ::= (<int_number> | <var_name>) ('weeks' | 'week')
<day_part> ::= (<int_number> | <var_name>) ('days' | 'day')
<hour_part> ::= (<int_number> | <var_name>) ('hours' | 'hour')
<minute_part> ::= (<int_number> | <var_name>) ('minutes' | 'minute' |
    'min')
<second_part> ::= (<int_number> | <var_name>) ('seconds' | 'second' |
    'sec')
<millisecond_part> ::= (<int_number> | <var_name>) ('milliseconds' |
    'millisecond' | 'msec')

<date_time> ::= [<dd> '/' <mm> '/' <aaaa>] ' ' <hh> 0*3(':'<ms>)

<dd> ::= DIGIT | ('1'DIGIT) | ('2'DIGIT) | '30' | '31'
<mm> ::= DIGIT | '10' | '11' | '12'
<aaaa> ::= 4DIGIT
<hh> ::= DIGIT | ('1'DIGIT) | '20' | '21' | '22' | '23'
<ms> ::= DIGIT | ('2'DIGIT) | ('3'DIGIT) | ('4'DIGIT) | ('5'DIGIT)
;ms means minute, seconds or milliseconds

```

```

<space_object> ::= +( '(' <latitude> ',' <longitude> ')' )

<latitude> ::= <lat_degree>'°' <ms>"" <ms>'"'(N|S)
              | ('-'|'+') <lat_degree> ['. ' *7DIGIT]
<lat_degree> ::= DIGIT | ('1'|'2'|'3'|'4'|'5'|'6'|'7'|'8')DIGIT | '90'

<longitude> ::= <long_degree>'°' <ms>"" <ms>'"'((W|E)|(O|E))
              | ('-'|'+') <long_degree> ['. ' *7DIGIT]
<long_degree> ::= DIGIT | 2DIGIT
                | '1'('0'|'1'|'2'|'3'|'4'|'5'|'6'|'7')DIGIT | '180'

<pattern_name> ::= <string>

<var_name> ::= <string>

<quant> ::= ∃ | ∀

<biconnector> ::= ∧ | ∨

<number> ::= <int_number> ['. '+DIGIT]

<int_number> ::= +DIGIT

<string> ::= ALPHA *(ALPHA | DIGIT | '_' )

<length_unit> ::= 'km' | 'm' | 'cm' | 'mi' | 'ft' | 'in'

```

Árvore Sintática

Dada essa gramática, a Figura 3.6 mostra árvore sintática do padrão *ex*, evidenciando a estrutura descrita nesta seção.

$$ex = x.value > 35 \vee x.value < 15 [1hour] \quad (3.1)$$

3.4.2 Semântica dos Elementos da Linguagem

A formação de padrão, na estrutura da linguagem descrita, é composta por um nome para o padrão, seguido de um conjunto de especificações de eventos que devem ocorrer dentro de uma janela de tempo. A janela temporal estabelece um corte sobre o fluxo de dados restringindo a execução do padrão a esse corte. O padrão combina as especificações de

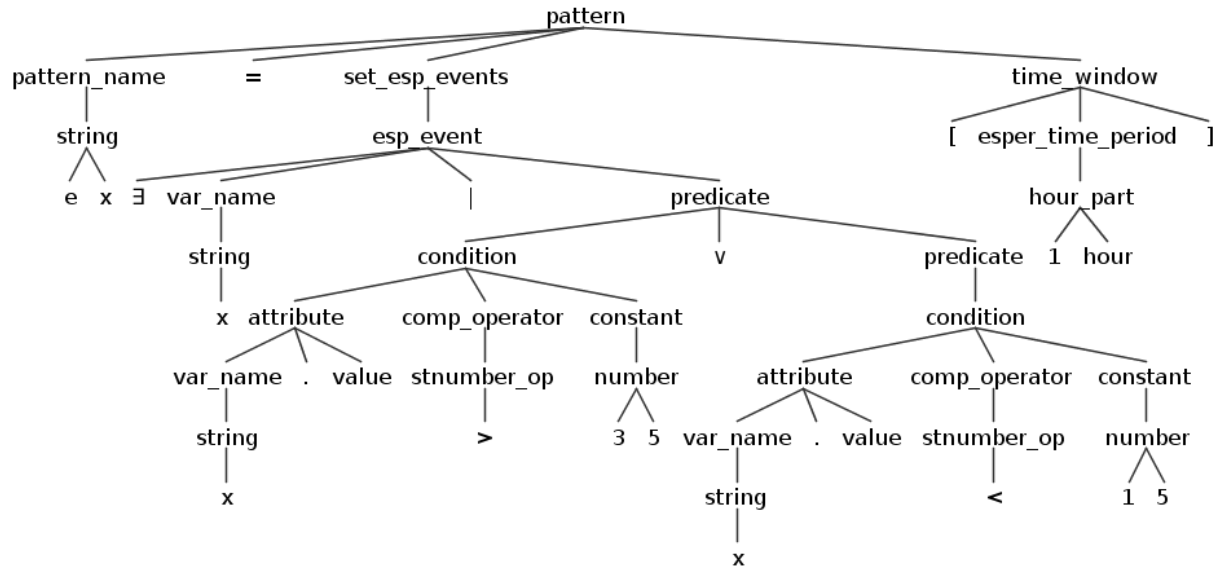


Figura 3.6: Árvore sintática do padrão *ex* gerada com a ferramenta ANTLR.

eventos e outros padrões (subpadrões), usando os conectores binários (\wedge e \vee) e unário (\neg). Cada especificação de eventos é formada por um quantificador (\exists , \forall) e um predicado combinado também por conectores. Os predicados são constituídos de relações entre os atributos dos eventos e constantes. As relações são estabelecidas pelo uso de operadores comparativos, que incluem operadores padrão ($=$, \neq , $>$, \geq , $<$, \leq) sobre atributos numéricos e literais, operadores temporais (e.g. *before*, *meet*, *during*, *equal*) e espaciais (e.g. *inside*, *overlap*, *disjoint*, *touch*).

Operadores Comparativos

Os operadores comparativos são aplicados sobre os atributos do evento. Operadores numéricos são aplicados sobre o atributo *value*; operadores literais, aplicados sobre os atributos *id*, *source*, *description* *value* e *unitOfMeasure*; operadores temporais, aplicados sobre os atributos *time*, *startValidTime* e *endValidTime*; e operadores espaciais, aplicados sobre o atributo *space*. Os operadores são da forma tradicional ou agrupada. A forma tradicional é binária, com as seguintes estruturas: *at op const* ou *at op at*. Exemplos de operadores aplicados na forma tradicional são:

$$x.value = 35 \quad (3.2)$$

$$x.value > y.value \quad (3.3)$$

A forma agregada visa simplificar a escrita de predicados longos que usam o mesmo operador e mesmo atributo para diferentes eventos. Ela possui duas estruturas: (*at1*, *at2*,

...) *op const/at* ou *op(at1, at2, ...)*. Na primeira forma, cada atributo apresentado na lista se relaciona com o último elemento (constante ou atributo) via o operador especificado. Exemplos são:

$$(x.value, y.value, z.value) > 35 \quad (3.4)$$

$$(x.value, y.value, z.value) > w.value \quad (3.5)$$

Nos exemplos, a forma agregada indica que os valores de x , y e z são maiores que 35, para o primeiro caso (exemplo 3.4); ou maiores que o valor de w , para o segundo caso (exemplo 3.5). A semântica agregada equivale à aplicação tradicional do operador para cada atributo na lista. No exemplo 3.4 a aplicação tradicional correspondente é: $(x.value > 35) \wedge (y.value > 35) \wedge (z.value > 35)$.

Na segunda forma agregada, a semântica da notação depende do operador. Os operadores podem adotar a semântica *total* ou *sequencial*. Na semântica total, a relação vale para todos os atributos na lista. Na sequencial, a relação se dá em sequência para cada par de atributos em sucessão.

Nesta linguagem, os operadores $=$ e \neq adotam a semântica total; já os operadores $>$, $<$, \geq e \leq adotam a semântica sequencial. Dentre os operadores temporais, *overlap* e *equal* adotam a semântica total; os demais (*before*, *meet*, *finish*, *during* e *start*) adotam semântica sequencial. Dentre os operadores espaciais, os operadores topológicos *disjoint*, *overlap* e *touch* têm semântica total; enquanto *inside* e *cross* são sequenciais; os operadores de orientação (*north* e *east*) têm semântica sequencial. As demais particularidades dos operadores espaciais e temporais serão explicadas posteriormente.

Exemplos da segunda forma agregada são:

$$\neq (x.value, y.value, z.value) \quad (3.6)$$

$$before(x.startValidTime, y.startValidTime, z.startValidTime) \quad (3.7)$$

O primeiro predicado exemplifica um operador agregado de semântica total (\neq), indicando que todos os valores (x , y e z) devem ser diferentes. Isto equivale à aplicação tradicional do operador para cada par de atributos. O segundo predicado exemplifica um operador agregado sequencial: o tempo inicial de x ocorre antes de y que ocorre antes de z . Isto equivale à aplicação tradicional: $(x \text{ before } y) \wedge (y \text{ before } z)$.

Em outras palavras, na sintaxe, operadores agrupados simplificam a forma de escrever um predicado. Esse é um diferencial da pesquisa, pois melhor evidencia as relações entre os eventos (contribuição (A) mencionada no início do capítulo).

A construção de predicados na linguagem permite, também, o uso de operadores aritméticos básicos ($+$, $-$, \times e \div) combinados com operadores comparativos. Assim,

elementos numéricos dos eventos podem participar de predicados do tipo:

$$x.value \times 2 > y.value \quad (3.8)$$

$$= (x.value, y.value + 1, z.value + 2) \quad (3.9)$$

Quantificadores

Os quantificadores \forall e \exists são utilizados no início de uma especificação de eventos para limitar a quantidade de eventos suficiente para validar o predicado.

- Quantificador Universal (\forall): todos os eventos presentes no fluxo devem satisfazer ao predicado especificado. Exemplos de padrões simples usando \forall são:

$$quant_ex1 : \forall x \quad (3.10)$$

$$quant_ex2 : \forall x | x.value > 35 \quad (3.11)$$

$$quant_ex3 : \forall x (x.description = "temp") | x.value > 35 \quad (3.12)$$

No exemplo 3.10, nenhum predicado foi especificado; portanto, o padrão *quant_ex1* retorna o fluxo inteiro, ou seja todos os eventos. O padrão 3.11 é satisfeito desde que todos os eventos do fluxo tenham valor maior que 35. O padrão 3.12 é satisfeito desde que todos os eventos relacionados à temperatura tenham o atributo *value* maior que 35.

- Quantificador Existencial (\exists): pelo menos um evento no fluxo deve satisfazer ao predicado especificado. Exemplos de padrões simples usando \exists são:

$$quant_ex4 : \exists x \quad (3.13)$$

$$quant_ex5 : \exists x | x.value > 35 \quad (3.14)$$

Em 3.13, o padrão é satisfeito pelo primeiro evento no fluxo, qualquer que seja. Em 3.14, o padrão é satisfeito por qualquer evento com valor maior que 35.

Conectores

Os conectores binários (\wedge e \vee) unário (\neg) são utilizados para combinar predicados ou especificações de eventos. Exemplos de padrões usando conectores para combinar predicados são:

$$con_ex1 : \exists x | x.description = "temp" \wedge x.value > 35 \quad (3.15)$$

$$con_ex2 : \exists x | x.value > 35 \vee x.value < 15 \quad (3.16)$$

$$con_ex3 : \exists x | x.value > 35 \vee (x.value < 15 \wedge x.description = "temp") \quad (3.17)$$

$$con_ex4 : \exists x | \neg(x.value > 35 \wedge x.description = "temp") \quad (3.18)$$

O padrão *con_ex1* representa a busca por um cenário que tenha ocorrido temperatura maior 35. O padrão *con_ex2* busca por eventos com valor maior que 35 ou menor que 15. Se, executados sobre um fluxo de dados de apenas temperatura, o padrão poderia representar um cenário de interesse por temperaturas extremas. O padrão *con_ex3* busca por um evento com temperatura menor que 15 ou qualquer evento com valor maior 35. Por fim, o padrão *con_ex4* busca por qualquer evento que não tenha temperatura maior que 35.

Exemplos de padrões que usam conectores para combinar especificações de eventos são:

$$con_ex5 : \exists x|x.value=35 \wedge \exists y|y.value=30 \quad (3.19)$$

$$con_ex6 : \forall x \wedge \exists y (y.id \neq x.id)|y.value=x.value \quad (3.20)$$

$$con_ex7 : \exists x|x.value > 35 \vee (\exists y|y.source="1" \wedge \exists z|z.value < y.value) \quad (3.21)$$

$$con_ex8 : \neg \exists |x.value > 35 \quad (3.22)$$

O primeiro padrão é satisfeito pela ocorrência de dois eventos no fluxo, um com valor 35 e outro com valor 30. O segundo (*con_ex6*) é satisfeito desde que para cada evento no fluxo exista pelo menos outro evento de mesmo valor. Isto é, o padrão é satisfeito desde que não exista um evento com valor único no fluxo. O padrão *con_ex7* é satisfeito pela existência de um evento com valor maior que 35 ou por dois eventos, desde que o primeiro tenha sido gerado pela fonte “1” e o segundo tenha valor menor que o valor do primeiro. Por fim, no padrão *con_ex8*, não existe um conjunto de eventos que satisfaça o padrão. Nesse caso, o padrão é satisfeito pela inexistência de evento com valor maior que 35 no fluxo analisado.

Operadores temporais

Os operadores temporais suportados pela linguagem são os operadores de um subconjunto simplificado dos métodos da ferramenta Esper [35] e utilizam as relações de Allen [4] (*before*, *meet*, *overlap*, *finish*, *during*, *start* e *equal*). O trabalho define 7 das 13 relações de Allen, de forma que as relações simétricas são incluídas na relação principal. Por exemplo, a relação *meet* inclui a semântica da relação *met_by*. Allen define essas relações aplicadas a um intervalo de tempo; para a adoção dessa semântica os operadores devem ser aplicados sobre o atributo *time* do evento, que representa um intervalo. No entanto, permitimos também o uso de alguns desses operadores sobre pontos, aplicando os operadores sobre os atributos *startValidTime* ou *endValidTime*. Os operadores que podem adotar semântica de pontos são apenas: *before* e *equal*.

A linguagem implementada no Esper adota opcionalmente a aplicação de 2 períodos de tempo como limiares inferiores e superiores para as relações. Inspirada nessa estrutura,

nossa proposta prevê o uso opcional de um período de tempo constante estabelecendo uma tolerância temporal para que o predicado seja satisfeito. A semântica desse limiar depende de cada relação, sendo abordada em cada operador apresentado a seguir. As ilustrações desta seção representa esse período de tempo por Δ .

- *before*[período de tempo]: expressa a relação temporal de precedência entre dois intervalos ou pontos, indicando um período mínimo de antecedência entre os elementos. Dados dois intervalos *a* e *b*, e dois pontos *c* e *d*:

$$a \text{ before } b \Rightarrow b.startValidTime - a.endValidTime > 0 \quad (3.23)$$

$$c \text{ before } d \Rightarrow d - c > 0 \quad (3.24)$$

$$a \text{ before[período]} b \Rightarrow b.startValidTime - a.endValidTime \geq \text{período} \quad (3.25)$$

$$c \text{ before[período]} d \Rightarrow d - c \geq \text{período} \quad (3.26)$$

A Figura 3.7 ilustra todas as definições.

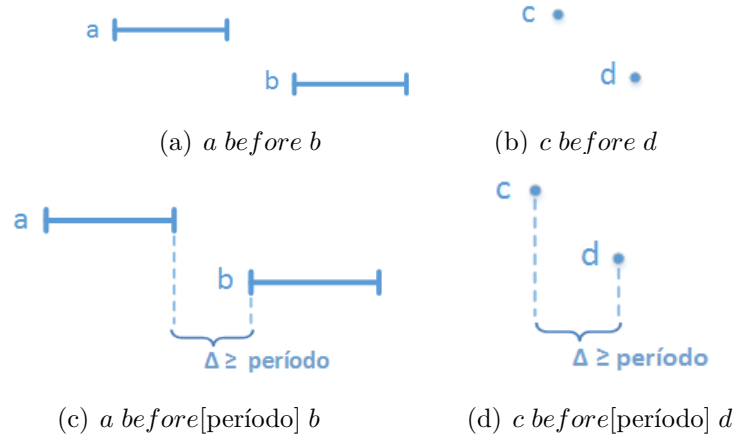


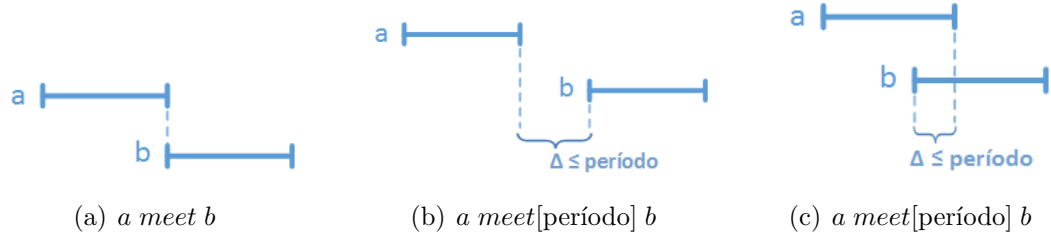
Figura 3.7: Semântica do operador *before* entre intervalos e pontos.

- *meet*[período de tempo]: expressa a relação temporal em que um intervalo finaliza no tempo inicial de outro. Quando especificado um período de tempo, esse representa uma tolerância no ponto de encontro dos intervalos. Dados dois intervalos *a* e *b*:

$$a \text{ meet } b \Rightarrow a.endValidTime = b.startValidTime \vee b.endValidTime = a.startValidTime \quad (3.27)$$

$$a \text{ meet[período]} b \Rightarrow |a.endValidTime - b.startValidTime| \leq \text{período} \vee |b.endValidTime - a.startValidTime| \leq \text{período} \quad (3.28)$$

A Figura 3.8 ilustra as definições.

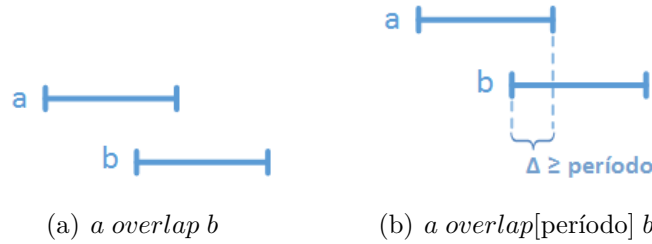
Figura 3.8: Semântica do operador *meet* entre intervalos.

- *overlap*[período de tempo]: expressa a relação em que intervalos de tempo são parcialmente sobrepostos. Quando especificado um período de tempo, esse representa um período mínimo de duração da parte sobreposta. Dados dois intervalos a e b :

$$a \text{ overlap } b \Rightarrow (a.startValidTime < b.startValidTime < a.endValidTime < b.endValidTime) \vee (b.startValidTime < a.startValidTime < b.endValidTime < a.endValidTime) \quad (3.29)$$

$$a \text{ overlap}[\text{período}] b \Rightarrow (a.startValidTime < b.startValidTime < a.endValidTime < b.endValidTime \wedge a.endValidTime - b.startValidTime \geq \text{período}) \vee (b.startValidTime < a.startValidTime < b.endValidTime < a.endValidTime \wedge b.endValidTime - a.startValidTime \geq \text{período}) \quad (3.30)$$

A Figura 3.9 ilustra as definições.

Figura 3.9: Semântica do operador *overlap* entre intervalos.

- *finish*[período de tempo]: expressa a relação em que dois intervalos começam em instantes distintos, mas terminam ao mesmo tempo. Quando especificado um período de tempo, esse representa uma tolerância no tempo final dos intervalos. Dados

dois intervalos a e b :

$$a \text{ finish } b \Rightarrow b.startValidTime \neq a.startValidTime \wedge \quad (3.31)$$

$$a.endValidTime = b.endValidTime$$

$$a \text{ finish[período]} b \Rightarrow b.startValidTime \neq a.startValidTime \wedge \quad (3.32)$$

$$|a.endValidTime - b.endValidTime| \leq \text{período}$$

A Figura 3.10 ilustra as definições.

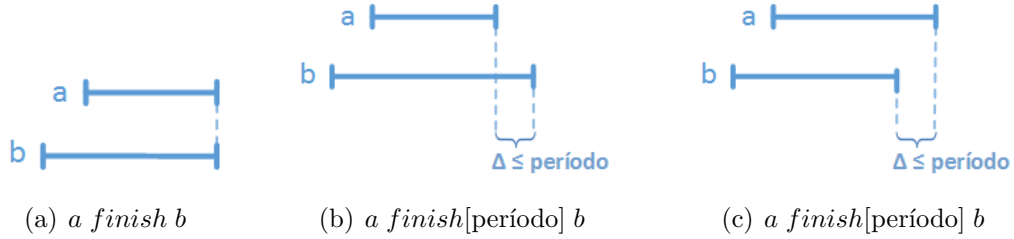


Figura 3.10: Semântica do operador *finish* entre intervalos.

- *during*[período de tempo]: expressa a relação em que um intervalo de tempo ocorre inteiramente durante outro intervalo. Quando especificado um período de tempo, esse representa uma tolerância na diferença de início e final dos intervalos. Dados dois intervalos a e b :

$$a \text{ during } b \Rightarrow b.startValidTime < a.startValidTime \leq \quad (3.33)$$

$$a.endValidTime < b.endValidTime$$

$$a \text{ during[período]} b \Rightarrow 0 < a.startValidTime - b.startValidTime \leq \text{período} \quad (3.34)$$

$$\wedge 0 < b.endValidTime - a.endValidTime \leq \text{período}$$

A Figura 3.11 ilustra as definições.

- *start*[período de tempo]: expressa a relação em que dois intervalos se iniciam ao mesmo tempo. Quando especificado um período de tempo, esse representa uma tolerância no tempo inicial dos intervalos. Dados dois intervalos a e b :

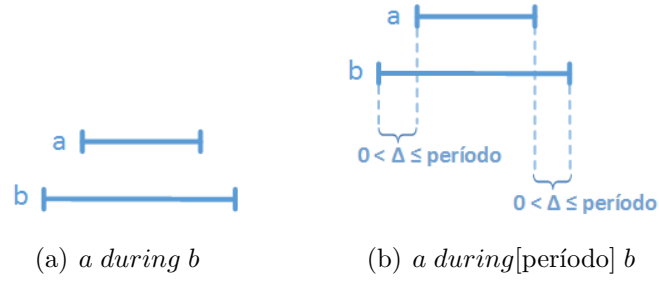
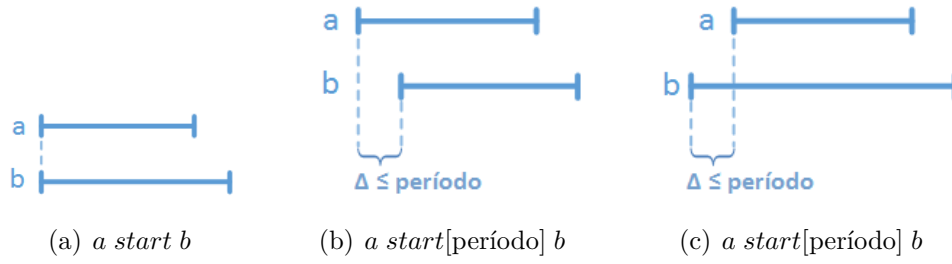
$$a \text{ start } b \Rightarrow a.endValidTime \neq b.endValidTime \wedge \quad (3.35)$$

$$a.startValidTime = b.startValidTime$$

$$a \text{ start[período]} b \Rightarrow a.endValidTime \neq b.endValidTime \wedge \quad (3.36)$$

$$|a.startValidTime - b.startValidTime| \leq \text{período}$$

A Figura 3.12 ilustra as definições.

Figura 3.11: Semântica do operador *during* entre intervalos.Figura 3.12: Semântica do operador *start* entre intervalos.

- *equal*[período de tempo]: expressa a relação de igualdade entre dois intervalos ou pontos. Quando especificado um período, esse representa uma tolerância entre os intervalos ou pontos. Dados dois intervalos a e b e dois pontos c e d :

$$a \text{ equal } b \Rightarrow a.startValidTime = b.startValidTime \wedge a.endValidTime = b.endValidTime \quad (3.37)$$

$$c \text{ equal } d \Rightarrow c = d \quad (3.38)$$

$$a \text{ equal}[\text{período}] b \Rightarrow |a.startValidTime - b.startValidTime| \leq \text{período} \quad (3.39)$$

$$\wedge |b.endValidTime - b.endValidTime| \leq \text{período}$$

$$c \text{ equal}[\text{período}] d \Rightarrow |d - c| \leq \text{período} \quad (3.40)$$

A Figura 3.13 ilustra essas as definições.

Operadores espaciais

O suporte a relações espaciais entre os eventos na especificação e detecção de padrões é um diferencial da pesquisa, pois permite a construção de cenários mais elaborados em biodiversidade (contribuição (B) da linguagem, mencionada no início do capítulo). A linguagem se baseia em relacionamentos espaciais consolidados na literatura de bancos de

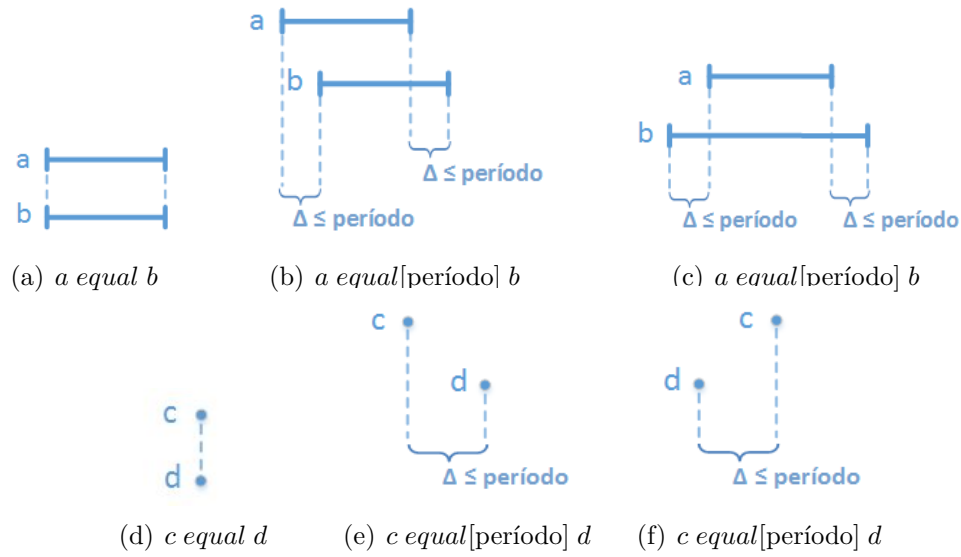
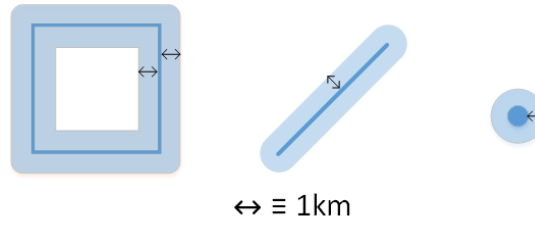


Figura 3.13: Semântica do operador *equal* entre intervalos e pontos.

dados espaciais ([13, 7, 8]) e espaço-temporais ([12, 29]). Guting [13] classifica as relações espaciais em relações topológicas, direcionais (ou de direção) e métricas. Na linguagem, relações topológicas e direcionais, por retornarem valor lógico, são tratadas por operadores espaciais; e relações métricas, por retornarem um valor escalar, são tratadas por funções espaciais.

A noção de tolerância no tempo é agora estendida para tolerância espacial. Para cada operador espacial, a tolerância é definida considerando a área de *buffer* gerada ao redor dos objetos espaciais. O parâmetro *buf* (extensão definida pelo especialista), agora, irá definir o tamanho do *buffer* (e não necessariamente a distância efetiva entre os elementos). Para ressaltar esse parâmetro, as expressões espaciais com *buffer* contêm o parâmetro [*buf*]. A Figura 3.14 exemplifica um *buffer* criado com extensão igual a 1km, sendo que o objeto poligonal tem *buffers* interno e externo. Em geral, usa-se apenas o *buffer* externo. Nesta dissertação, o interno apenas será utilizado quando dito explicitamente, caso contrário, apenas *buffer* externo é gerado.

Na linguagem especificada, operadores espaciais são aplicados sobre o atributo *space* dos eventos. Esse atributo possui múltiplos objetos espaciais representados por pontos, linhas e polígonos. Para efeito dos operadores espaciais, todos os predicados são avaliados sobre o MBR (*Minumum Bounding Rectangle*) envolvendo a(s) geometria(s) do atributo *space*, ou seja, se um atributo *space* tiver mais de um objeto espacial, os operadores espaciais serão aplicados sobre o MBR que envolve o conjunto de objetos. A Figura 3.15 ilustra um exemplo. Os conjuntos de A e B são envolvidos por dois MBR e qualquer

Figura 3.14: Objetos com *buffer* de 1km.

operador espacial $SP(A, B)$ é calculado da seguinte forma

$$SP(A, B) = SP(MBR(A), MBR(B)).$$

Da mesma forma, se usado um *buffer* associado a objetos de A e B, então o *buffer* é aplicado aos MBR correspondentes:

$$SP[buf](A, B) = SP[buf](MBR(A), MBR(B)).$$

Dessa forma, os operadores a seguir são definidos por sua aplicação sobre um único objeto espacial, seja ele um ponto, linha ou polígono (inclusive o polígono que representa o MBR de um conjunto de objetos).

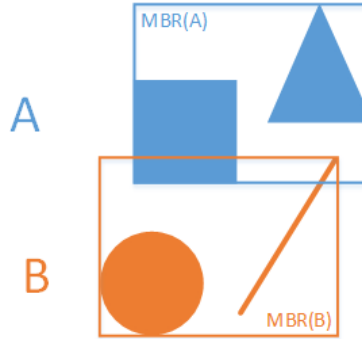


Figura 3.15: Operadores espaciais sobre conjuntos de objetos são aplicados sobre os MBR dos conjuntos.

Operadores espaciais de direção.

Os operadores direcionais indicam a direção relativa entre as disposições dos objetos geométricos no espaço. Os operadores suportados são o conjunto mínimo: *north* e *east*. Outras relações podem ser derivadas dessas. As definições dos operadores a seguir são ligeiramente adaptadas do trabalho de Faria [12] e adicionam o suporte à especificação de um *buffer* com extensão definida em $[buf]$ como tolerância na relação.

A relação $north[buf]$ expressa a relação em que um objeto espacial está localizado inteiramente a norte de outro objeto. A relação $east[buf]$ expressa a relação em que um objeto espacial está localizado inteiramente a leste de outro objeto. Em ambas as relações, as geometrias do *buffer* de cada objeto envolvido são construídas usando o parâmetro *buf*. Dados objetos espaciais a e b ; dois pontos a_i e b_i pertencentes às geometrias objeto a e b , respectivamente; dois objetos espaciais a^b e b^b compostos pela geometria de *buffer* gerada sobre os objetos a e b , respectivamente; e dois pontos a_i^b e b_i^b pertencentes às geometrias de a^b e b^b , respectivamente:

$$a \text{ north } b \Rightarrow \forall a_i \in a \forall b_i \in b \mid latitude(a_i) \geq latitude(b_i) \quad (3.41)$$

$$a \text{ north}[buf] b \Rightarrow \forall a_i^b \in a^b \forall b_i^b \in b^b \mid latitude(a_i^b) \geq latitude(b_i^b) \quad (3.42)$$

$$a \text{ east } b \Rightarrow \forall a_i \in a \forall b_i \in b \mid longitude(a_i) \geq longitude(b_i) \quad (3.43)$$

$$a \text{ east}[buf] b \Rightarrow \forall a_i^b \in a^b \forall b_i^b \in b^b \mid longitude(a_i^b) \geq longitude(b_i^b) \quad (3.44)$$

A Figura 3.16 ilustra possíveis disposição de objetos usando o operador $north$. E a Figura 3.17 ilustra possíveis disposição de objetos usando o operador $east$. A letra (c) evidencia um caso onde a está localizado a nordeste de b .

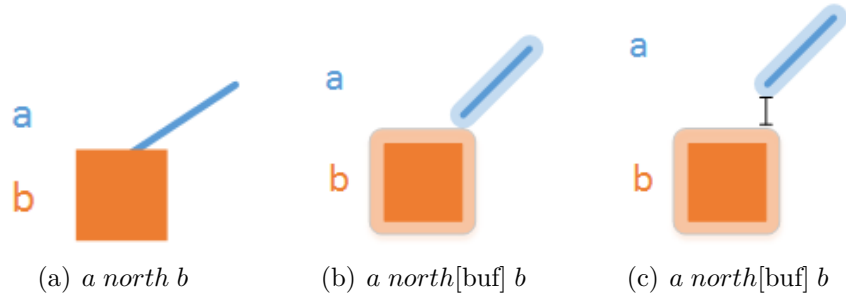


Figura 3.16: Semântica do operador $north$ entre dois objetos a e b .

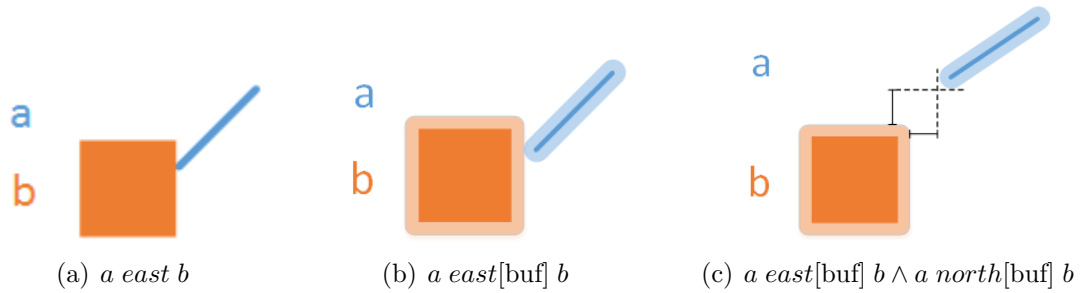


Figura 3.17: Semântica do operador $east$ entre dois objetos a e b .

Operadores espaciais topológicos

Os operadores espaciais topológicos suportados pela linguagem são as relações topológicas mínimas propostas por Clementini et al. [8] (*disjoint*, *touch*, *overlap*, *inside* e *cross*). As definições a seguir adotam a formalização clássica utilizada por Clementini et al. [8], Câmara et al. [7] e, posteriormente, simplificada por Faria [12]. A notação utilizada é ligeiramente modificada (de Ω e ω para A e a , respectivamente) e sua semântica apenas é acrescida para suportar a especificação de um *buffer* como tolerância na relação.

No espaço \mathbb{R}^2 , um objeto topológico simples a é classificado em três tipos:

1. Ponto;
2. Linha que não se intersecta, podendo ser circular ou possuir apenas dois pontos terminais;
3. Região simples conectada e sem buracos, ou seja, não é a união de conjuntos disjuntos de pontos.

A dimensão de um conjunto de objetos topológicos simples A é dada por:

$$\dim(A) = \begin{cases} - & \text{se } A = \emptyset \\ 0 & \text{se } A \text{ contém pelo menos um ponto e nenhuma linha ou região} \\ 1 & \text{se } A \text{ contém pelo menos uma linha e nenhuma região} \\ 2 & \text{se } A \text{ contém pelo menos uma região} \end{cases}$$

A fronteira de objeto topológico simples a , denotada por δa , é dada por:

$$\delta a = \begin{cases} \emptyset & \text{se } a \text{ é um ponto ou linha circular} \\ \{P, Q\} & \text{se } a \text{ é uma linha não circular e } P \text{ e } Q \text{ são seus pontos terminais} \\ L & \text{se } a \text{ é uma região simples e } L \text{ é a linha circular formada por todos os pontos de acumulação de } a \end{cases}$$

O interior de um objeto topológico a , denotado por a^0 , é definido como:

$$a^0 = a - \delta a$$

Em adição à definição clássica, o *buffer* de um objeto topológico a , denotado por a^b para *buffer* externo e por a^{bi} para *buffer* interno (aplicado apenas em regiões), é um objeto topológico de tipo região simples. As definições de dimensão, fronteira e interior também podem ser aplicadas aos *buffer*.

- *disjoint* e *disjoint*[buf]: representa a relação em que os objetos topológicos envolvidos são disjuntos. Quando *buf* é especificado como tolerância, a relação *disjoint* deve ser aplicada sobre as geometrias de *buffer* (com extensão *buf*) de cada objeto topológico. Dessa forma, para quaisquer dois objetos topológicos *a* e *b*:

$$a \text{ disjoint } b = (a \cap b = \emptyset) \quad (3.45)$$

$$a \text{ disjoint[buf] } b = (a^b \cap b^b = \emptyset) \quad (3.46)$$

As Figuras 3.18 e 3.19 ilustram os relacionamentos *disjoint* e *disjoint*[buf] entre 2 objetos topológicos.

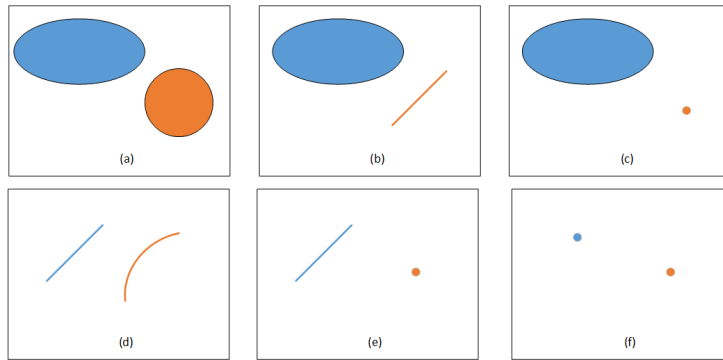


Figura 3.18: Relacionamento *disjoint* entre objetos topológicos.

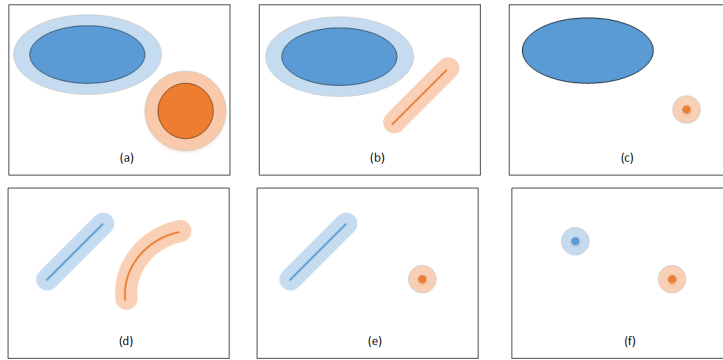


Figura 3.19: Relacionamento *disjoint*[buf] entre objetos topológicos.

- *touch* e *touch*[buf]: representa a relação em que os objetos topológicos envolvidos se tocam. Quando *buf* é especificado como tolerância, a relação *touch* é satisfeita desde que as geometrias de *buffer* (com extensão *buf*) dos objetos topológicos considerados tenham alguma intersecção. Dessa forma, para dois objetos dos tipos região e região,

linha e região, linha e linha, ponto e região, e ponto e linha (exceto ponto e ponto):

$$a \text{ touch } b = (a \cap b \neq \emptyset) \wedge (a^0 \cap b^0 = \emptyset) \quad (3.47)$$

$$a \text{ touch[buf] } b = (a \cap b = \emptyset) \wedge (a^0 \cap b^0 = \emptyset) \wedge (a^b \cap b^b \neq \emptyset) \quad (3.48)$$

As Figuras 3.20 e 3.21 ilustram os relacionamentos *touch* e *touch[buf]* entre 2 objetos topológicos.

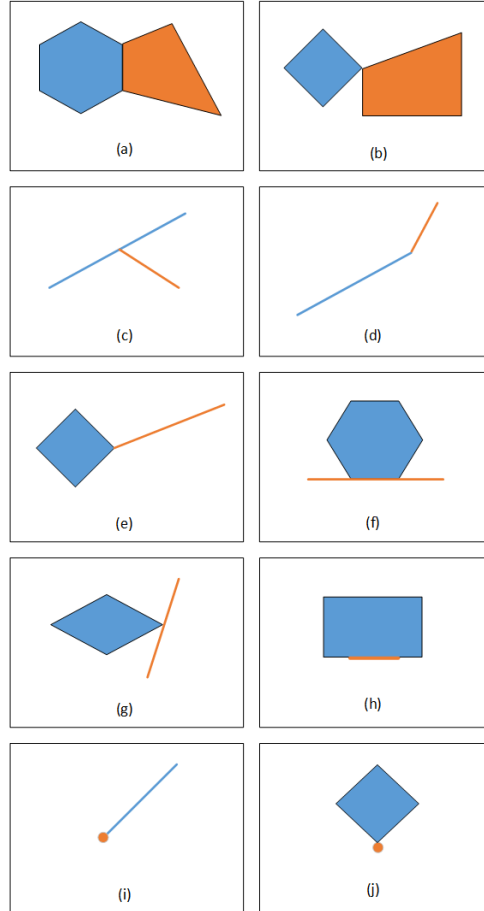


Figura 3.20: Relacionamento *touch* entre objetos topológicos.

- *overlap* e *overlap[buf]*: representa a relação em que os objetos topológicos envolvidos têm parte sobreposta. Quando *buf* é especificado como tolerância, a relação *overlap* é satisfeita desde os objetos envolvidos sejam regiões e as geometrias formadas pela fronteira de seus *buffers* internos (com extensão *buf*) tenham parte

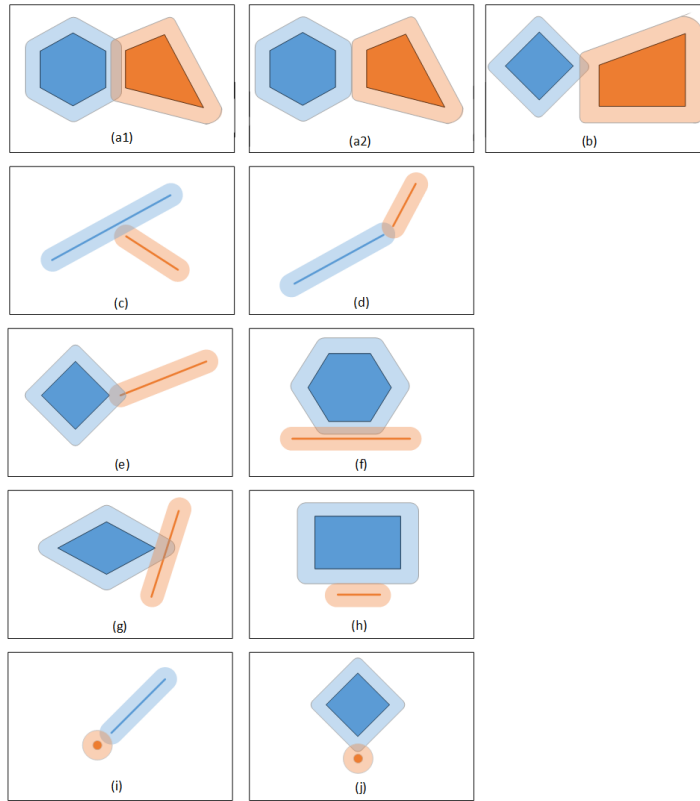


Figura 3.21: Relacionamento $touch[buf]$ entre objetos topológicos.

sobreposta. Dessa forma, para dois objetos dos tipos região e região, e linha e linha:

$$a \text{ overlap } b = (a \cap b \neq a) \wedge (a \cap b \neq b) \wedge \dim(a^0 \cap b^0) = \dim(a^0) = \dim(b^0) \quad (3.49)$$

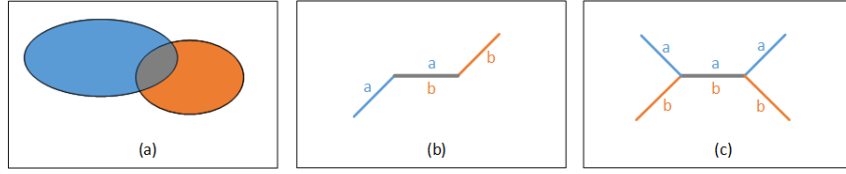
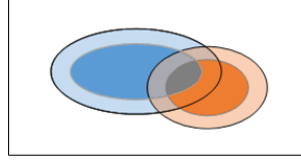
$$a \text{ overlap}[buf] b = (a \cap b \neq a) \wedge (a \cap b \neq b) \wedge \dim(a^0 \cap b^0) = \dim(a^0) = \dim(b^0) = \dim(a^0 \cap b^0 - a^{bi} - b^{bi}), \quad (3.50)$$

onde a e b são regiões

$$a \text{ overlap}[buf] b = a \text{ overlap } b, \text{ onde } a \text{ e } b \text{ são linhas}$$

As Figuras 3.22 e 3.23 ilustram o o relacionamento $overlap$ e $overlap[buf]$ entre 2 objetos topológicos.

- *inside* e *inside[buf]*: Dados dois objetos topológicos a e b , $a \text{ inside } b$ representa a relação em que a está dentro do objeto B . Quando buf é especificado como tolerância, a relação *inside* (aplicada apenas se b é uma região) é satisfeita desde que a geometria de *buffer* (com extensão buf) do objeto a esteja dentro do objeto

Figura 3.22: Relacionamento *overlap* entre objetos topológicos.Figura 3.23: Relacionamento *overlap[buf]* entre objetos topológicos.

b. Dessa forma, para quaisquer dois objetos:

$$a \text{ inside } b = (a^0 \cap b^0 \neq \emptyset) \wedge (a \cap b = a) \quad (3.51)$$

$$a \text{ inside[buf]} b = a \text{ inside } b, \text{ se } b \text{ é uma linha ou ponto} \quad (3.52)$$

$$a \text{ inside[buf]} b = (a^0 \cap b^0 \neq \emptyset) \wedge (a^b \cap b = a^b), \text{ se } b \text{ é uma região} \quad (3.53)$$

As Figuras 3.24 e 3.25 ilustram os relacionamentos *inside* e *inside[buf]* entre 2 objetos topológicos, onde *a* (em azul) está dentro de *b* (em laranja).

- *cross* e *cross[buf]*: Dados dois objetos topológicos *a* e *b*, *a cross b* representa a relação em que *a* cruza o objeto *b*. Nessa relação não se aplica tolerância. Dessa forma, para objetos do tipo linha e região, e linha e linha:

$$a \text{ cross } b = (a \cap b \neq a) \wedge (a \cap b \neq b) \wedge \dim(a^0 \cap b^0) = \quad (3.54)$$

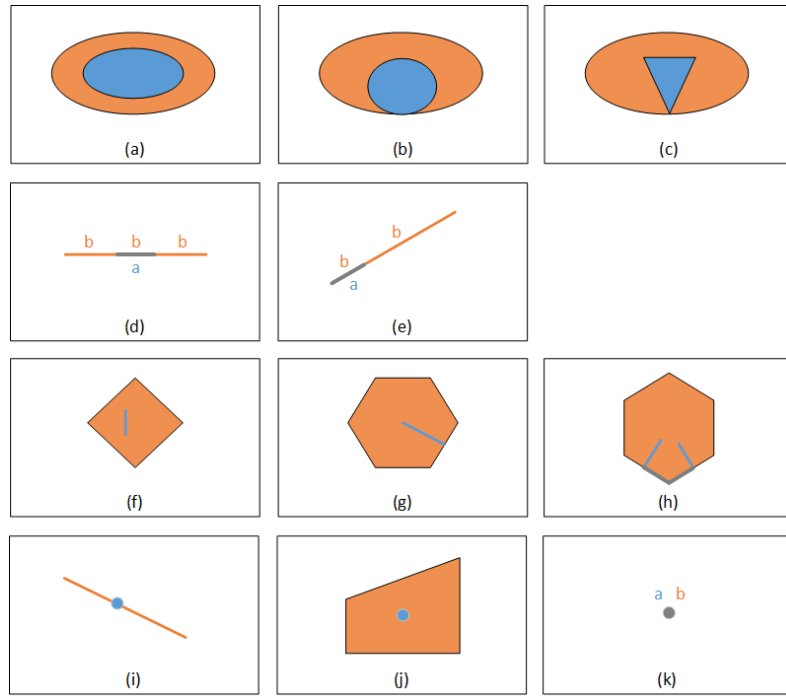
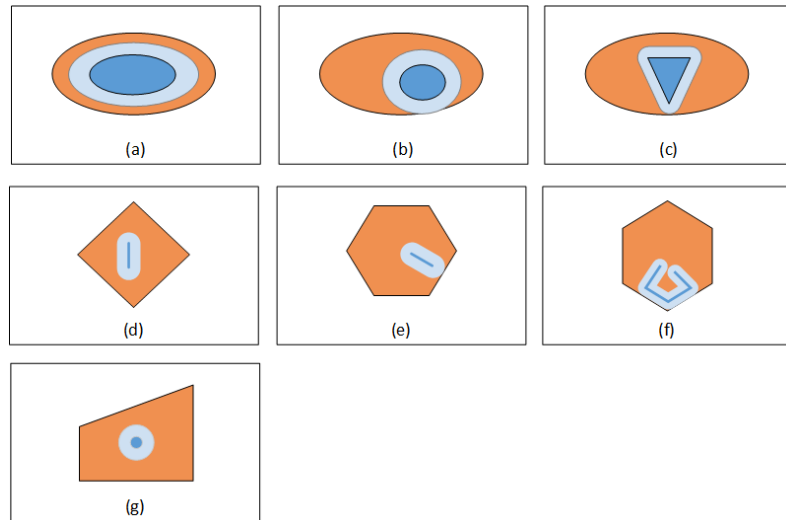
$$(\max(\dim(a^0), \dim(b^0)) - 1)$$

$$a \text{ cross[buf]} b = a \text{ cross } b \quad (3.55)$$

A Figura 3.26 ilustra o o relacionamento *cross* entre 2 objetos topológicos, onde *a* (em azul) cruza *b* (em laranja).

Funções espaciais

As funções espaciais implementam as relações métricas entre os objetos espaciais. A linguagem suporta as funções:

Figura 3.24: Relacionamento *inside* entre objetos topológicos.Figura 3.25: Relacionamento *inside[buf]* entre objetos topológicos.

- $area(A)$: Dado um conjunto de objetos espaciais A , $area(A)$ retorna o somatório da área de todos os objetos a_i em A , se a_i é uma região.

$$area(A) = \sum_{i=1}^{i=n} area(a_i \in A) , \text{ se } a_i \text{ é uma região} \quad (3.56)$$

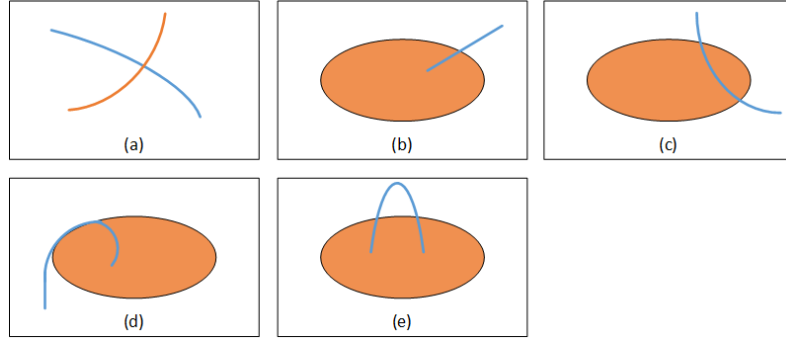


Figura 3.26: Relacionamento *cross* entre objetos topológicos.

- $length(A)$: Dado um conjunto de objetos espaciais A , $length(A)$ retorna o somatório dos comprimentos de todos os objetos a_i em A , se a_i é uma linha.

$$length(A) = \sum_{i=1}^{i=n} length(a_i \in A) , \text{ se } a_i \text{ é uma linha} \quad (3.57)$$

- $perimeter(A)$: Dado um conjunto de objetos espaciais A , $perimeter(A)$ retorna o somatório dos perímetros de todos os objetos a_i em A , se a_i é uma região.

$$perimeter(A) = \sum_{i=1}^{i=n} perimeter(a_i \in A) , \text{ se } a_i \text{ é uma região} \quad (3.58)$$

- $distance(A, B)$: Dados dois conjuntos de objetos espaciais A e B , $distance(A, B)$ retorna a distância mínima entre os MBRs das geometrias em A e B .

$$distance(A, B) = distance(MBR(A), MBR(B)) \quad (3.59)$$

Na linguagem, as funções espaciais são utilizadas na construção de predicados em combinação com operadores comparativos, como exemplo:

$$distance(x.space, y.space) > 10 \quad (3.60)$$

$$area(x.space) = area(y.space) \quad (3.61)$$

Janela temporal ([período de tempo])

A janela temporal limita a detecção de padrão a um subconjunto temporal do fluxo. A janela caminha sobre o fluxo de eventos ao longo do tempo, mantendo dentro dela apenas os eventos válidos no período especificado. Os eventos válidos são aqueles cujo tempo de validade final (*endValidTime*) está dentro da janela ou é posterior a ela.

Essa abordagem é ligeiramente modificada da literatura pela noção de validade dos eventos. Na literatura, a janela considera o tempo em que os eventos foram recebidos no fluxo, ao invés do seu tempo de validade, que na maioria das vezes nem existe. A razão dessa escolha é que o tratamento de eventos é assíncrono e fica mais fácil processá-los usando o tempo de chegada. No entanto, nossa abordagem busca por eventos que correspondam ao cenário atual, mesmo que a informação tenha sido coletada anteriormente ao intervalo da janela. Essa noção é importante pois permite o tratamento de dados de diversas naturezas e escalas. Por exemplo, permite que dados mensurados com baixa frequência e de natureza constante (em geral, com longo tempo de validade) sejam utilizados para representar o cenário atual. A Figura 3.27 ilustra esse conceito da janela temporal.

Nessa figura, a janela é representada pelo retângulo sombreado sobre um corte do fluxo de dados. Os eventos são estendidos ao longo do tempo de acordo com seu período de validade. Por exemplo, $e1.startValidTime = t - 1$ e $e1.endValidTime = t + 2$. Em cada parte da imagem é possível notar que a janela se estende do tempo atual até o passado de acordo com o seu tamanho (3 segundos). Portanto, em (a) o tempo atual é t e a janela compreende t até $t - 3$. Nesse passo e um passo seguinte (letras (a) e (b)), os eventos $e1$, $e2$, $e3$ e $e4$ participam da janela. No passo (c), o evento $e2$ deixa a janela. E 8 segundos após t (letra (d)), apenas o evento $e5$ participa da janela.

A janela é um operador temporal com tratamento diferente na linguagem; não possui forma binária e agregada e não necessita explicitar o atributo da operação (*time* composto por *startValidTime* e *endValidTime*). O operador é aplicado sobre todo o padrão, indicando que todos os eventos especificados devem ocorrer dentro dessa janela. A janela é definida por um período de tempo que, segundo a especificação da EPL Esper, explicita períodos em anos, meses, semanas, dias, horas, minutos ou segundos. O exemplo a seguir adiciona a janela de tempo a um padrão que busca por 3 eventos gerados por diferentes fontes.

$$timewindow_ex1 : \exists x \wedge \exists y \wedge \exists z | \neq (x.source, y.source, z.source)[30sec] \quad (3.62)$$

Subpadrões

Visando o reuso de padrões, a linguagem permite a composição hierárquica de padrões progressivamente mais complexos utilizando padrões já especificados (contribuição (C) da linguagem, mencionada no início da seção 3.4). O reuso de subpadrões permite a construção de cenários arbitrariamente complexos. Por exemplo, a especificação de um cenário de frente fria pode reutilizar a especificação de um cenário de chuva forte e combinar os demais fatores (essas construções mais complexas são vistas no capítulo 4). Os padrões a seguir exemplificam o reuso de subpadrões combinados por conectores; e a Figura 3.28

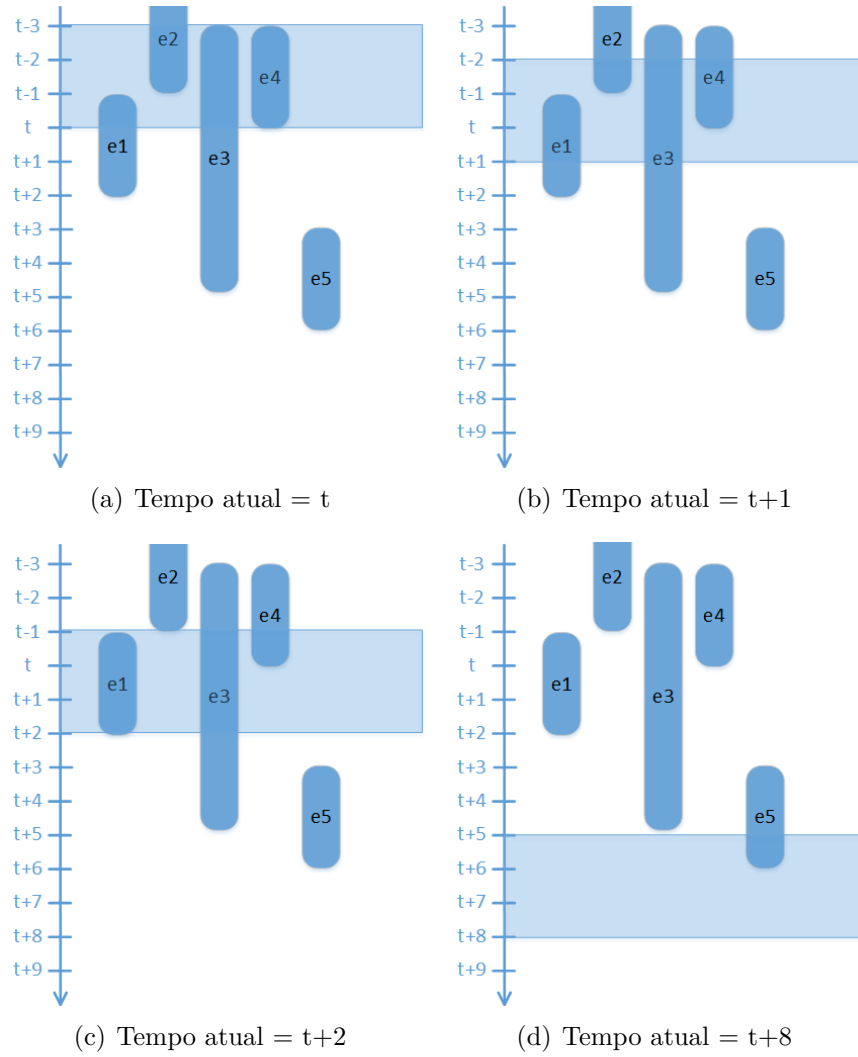


Figura 3.27: Eventos em uma janela temporal de 3 segundos.

evidencia a construção hierárquica. Esses exemplos pressupõem que todos os fenômenos ocorrem no mesmo local. Caso contrário, haveria necessidade de acrescentar predicados espaciais.

$$padraoex1 = \exists x \mid x.description = "temp" \wedge x.value > 28 \wedge chuva [30sec] \quad (3.63)$$

$$padraoex2 = chuva \wedge frio [1min] \quad (3.64)$$

$$chuva = \exists x \mid x.description = "chuva" \wedge x.value > 5 \quad (3.65)$$

$$frio = \exists x \mid x.description = "temp" \wedge x.value < 20 \vee vento [30sec] \quad (3.66)$$

$$vento = \exists x \mid x.description = "vento" \wedge x.value > 10 \quad (3.67)$$

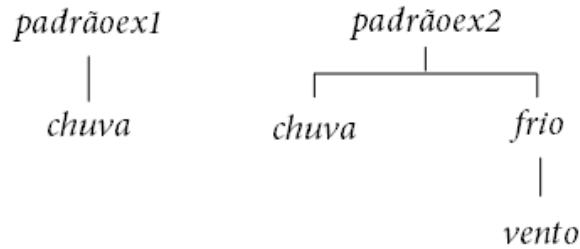


Figura 3.28: Exemplos de padrões hierárquicos.

Os padrões buscam pelos cenários: ocorrência de chuva em clima quente (*padraox1*) e ocorrência de chuva em clima frio (*padraox2*). Em geral, as janelas temporais de subpadrões são menores que as janelas dos padrões maiores (ou mais gerais). O contrário não necessariamente implica em construções incorretas ou em padrões que nunca são casados; no entanto, o casamento deve ocorrer dentro da janela especificada pelo padrão mais geral.

Capítulo 4

Estudo de Caso

Este estudo de caso mostra a aplicação da nossa proposta na detecção de fenômenos ambientais, que possam ser interessantes para análises no contexto de biodiversidade. No estudo, dados ambientais são representados por eventos simples de acordo com a estrutura definida na seção 3.2; cenários de interesse são representados por padrões escritos com a linguagem proposta na seção 3.4; e eventos complexos resultantes da execução do padrão são representados de acordo com a estrutura proposta em seção 3.3.

O cenário 1 mostra a hierarquia de eventos complexos e aplicação de operadores temporais, sendo que os cenários 2 e 3 adicionam relações espaciais ao cenário 1. O cenário 4 é um estudo de caso mais completo que explora: os aspectos temporais e espaciais do cenário; a composição de padrões usando subpadrões; e a aplicação de operadores de forma agregada, simplificando a especificação de eventos.

Para todos os cenários, iremos considerar os seguintes fluxos de dados:

- Dados meteorológicos gerados a cada 2 minutos: intensidade da chuva, temperatura, velocidade do vento, direção do vento, etc.
- Dados meteorológicos diários: temperatura máxima, temperatura mínima, acúmulo de chuva, etc.
- Mapa hidrográfico atualizado anualmente: representado por eventos com: *description* $\in \{\text{“rio”}, \text{“bacia”}, \text{“represa” e outros}\}$; *value* equivalente ao nome do rio, da bacia, da represa e assim por diante.

Cada evento tem uma localização associada. Na granularidade espacial considerada, cidades e regiões são descritas por polígonos; rios ou estradas são descritos por linhas; Estabelecimentos, nascentes de rio e demais referências pontuais são descritos por pontos. Para todos os estudos de casos, pressuõe-se que o processamento vai considerar eventos que chegem dentro de uma janela temporal, a menos que haja outras restrições impostas pelo padrão.

4.1 Cenário 1: Chuva forte

Para este cenário, adotamos a definição de chuva forte segundo o *Central Weather Bureau* [6]. Como visto na seção 3.3, uma chuva forte ocorre se o acúmulo diário (24h) de chuva excede 50mm, e, dentro das 24 horas, o acúmulo de chuva em 1 hora excede 15mm. A Tabela 4.1 apresenta parte do fluxo considerado para este cenário, com dados de intensidade de chuva (intChuva) a cada 2 minutos e acúmulo diário de chuva (acumChuvaD) para uma mesma localização (região R1). Para tais dados, o cenário de chuva forte pode ser representado pelos padrões:

$$acumChuvaHMaior15 = \forall a(a.description = "intChuva") | a.value \geq 15[1hour] \quad (4.1)$$

$$chuvaforte = \exists b | b.description = "acumChuvaD" \wedge b.value \geq 50 \wedge \quad (4.2)$$

$$\exists c | c.description = "acumChuvaHMaior15" \wedge$$

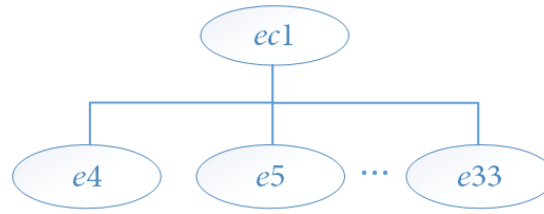
$$c.time \text{ during } b.time$$

Tabela 4.1: Eventos no corte *Fluxo1* do fluxo de eventos

id	source	description	value	unitOf Measure	time		space
					startValidTime	endValidTime	
e1	source1	"acumChuvaD"	60	mm	00:00 10/05/2015	00:00 11/05/2015	R1
e2	source1	"intChuva"	0	mm/h	00:00 10/05/2015	00:02 10/05/2015	R1
e3	source1	"intChuva"	10	mm/h	00:02 10/05/2015	00:04 10/05/2015	R1
e4	source1	"intChuva"	15	mm/h	00:04 10/05/2015	00:06 10/05/2015	R1
e5	source1	"intChuva"	17	mm/h	00:06 10/05/2015	00:08 10/05/2015	R1
...	e6 a e32 consecutivos com intensidade = 17mm/h a cada 2 minutos						
e33	source1	"intChuva"	17	mm/h	01:02 10/05/2015	01:04 10/05/2015	R1
e200	source1	"acumChuvaD"	70	mm	00:00 11/05/2015	00:00 12/05/2015	R1
e201	source1	"intChuva"	50	mm/h	00:00 11/05/2015	00:02 11/05/2015	R1
...	e202 a e214 consecutivos com intensidade = 50mm/h						
e215	source1	"intChuva"	0	mm/h	00:30 11/05/2015	00:32 11/05/2015	R1
...	e216 a e229 consecutivos com ausência de chuva (intensidade=0mm/h)						
e230	source1	"intChuva"	0	mm/h	00:58 11/05/2015	01:00 11/05/2015	R1

A ocorrência da sequencia de eventos entre *e4* e *e33* gera o evento *ec1* (que detecta que durante 1 hora foi constatada chuva $\geq 15mm$), dado o padrão *acumChuvaHMaior15*. A Figura 4.1 ilustra a visão hierárquica do evento. Ao mesmo tempo, são geradas as estruturas correspondentes, ilustradas nas Tabelas 4.2 e 4.3.

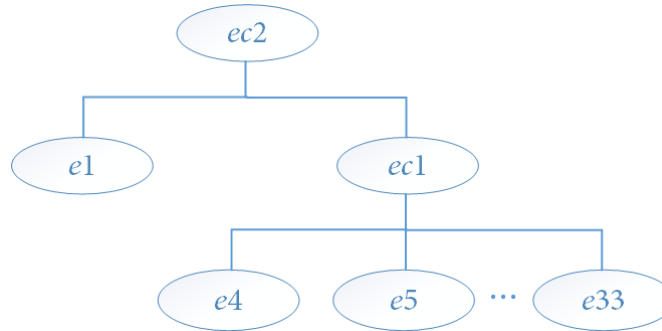
A verificação do padrão *chuvaforte* sobre o fluxo gera o evento *ec2* (simultaneidade de chuva acumulada durante 24 horas (*e1*) no mesmo período que *ec1*). A Figura 4.2 mostra uma possível construção de *ec2* e as Tabelas 4.4 e 4.5 detalham a construção, que é adicionada ao *Fluxo1*.

Figura 4.1: Evento Complexo gerado pela verificação do padrão *acumChuvaHMaior15*.Tabela 4.2: Evento *ec1* adicionado ao *Fluxo1*, parte 1

id	source	description	value	unitOf Measure	time		space
					startValidTime	endValidTime	
ec1	sistema	“acumChuvaHMaior15”	-	-	00:04 10/05/2015	01:04 10/05/2015	R1

Tabela 4.3: Evento *ec1* adicionado ao *Fluxo1*, parte 2

id	operatorList	eventList
ec1	<value, \geq , 15>, <description, =, “intChuva”>, <time, within, 1 hour>	e4, e5, ..., e33

Figura 4.2: Evento Complexo gerado pela verificação do padrão *chuvaforte*.Tabela 4.4: Estrutura do evento *ec2*, parte 1

id	source	description	value	unitOf Measure	time		space
					startValidTime	endValidTime	
ec2	sistema	“chuvaForte”	-	-	00:04 10/05/2015	01:04 10/05/2015	R1

Tabela 4.5: Estrutura do evento *ec2*, parte 2

id	operatorList	eventList
ec2	<time, <i>during</i> , time>	e1, ec1

Discussão

Esta discussão aponta aspectos positivos e negativos da nossa abordagem em relação à EPL da ferramenta Esper [35], bem consolidada na literatura. Utilizando Esper, o cenário de chuva forte também é escrito pela combinação de dois padrões. Sejam os mesmos atributos da Tabela 4.1, o primeiro calcula o acúmulo de chuva em 1 hora e o insere no fluxo de eventos, e o segundo busca pelo cenário de chuva forte:

```
insert into Fluxo1
select "acumChuvaH" as description, sum(value)/count(*) as value, space,
min(startValidtime) as starttime, max(endTime) as endValidTime
from fluxo(description="intChuva").win:time(1 hour)

select *
from Fluxo1 as B, Fluxo1 as C
where B.description="acumChuvaD", B.value>50,
C.description="acumChuvaH", C.value>15, C.during(B)
```

O ponto forte da nossa abordagem é a geração hierárquica de eventos complexos no processo de detecção de padrões. O resultado da detecção é um único evento que relaciona todos os eventos envolvidos no cenário. Esse evento é realimentado no fluxo para que possa ser usado por outro padrão. Por exemplo, o padrão *chuvaForte* usa eventos gerados por *acumChuvaHMaior15*. Esper apenas permite gerar eventos ao calcular valores agregados através de funções de agregação (*sum*, *count*, *max*, etc.), como no primeiro padrão. No entanto, esses eventos não possuem qualquer referência aos eventos que os compõem ou as operações usadas na composição. As demais construções de padrão em Esper não geram eventos, apenas retornam o conjunto de eventos que compõem o cenário.

Por focar no relacionamento entre eventos de diferentes naturezas, a linguagem proposta não permite o uso de funções agregadas (aplicadas apenas em eventos de mesma natureza). Além disso, muitas vezes, dados ambientais derivados requerem cálculos complexos, portanto já são pré-calculados e recebidos pelo sistema como dados primitivos. Porém, neste estudo de caso, não suportar agregação é o ponto fraco da nossa abordagem em composição. O uso de funções agregadas em Esper permite o cálculo do acúmulo de chuva em 1 hora. Isto leva à detecção de chuva forte também no dia 11 de maio, através evento *e200* e do acúmulo de chuva em *25mm* dos eventos *e201* a *e230*. O padrão descrito por nossa abordagem considera um número limitado de situações em que o acúmulo excede *15mm*, falhando em detectar chuva forte no dia 11.

Por fim, ambas as linguagens suportam relacionamentos temporais via operadores e janela temporal. Além disso, possibilitam o uso de tolerância nas relações.

4.2 Cenários 2 e 3: Chuva Forte Incluindo Aspectos Espaciais

A ocorrência de chuva forte pode ter diferentes impactos para diferentes regiões. Dessa forma, esta seção exemplifica dois cenários de chuva forte em regiões de interesse mais específicas. O cenário 2 representa chuva forte em regiões vizinhas e o cenário 3 representa chuva forte em regiões cortadas por, pelo menos, um rio.

A construção desses cenários acrescenta relações espaciais entre os eventos, permitindo considerar dados com diferentes localizações. Assim, o primeiro passo é estender os padrões anteriores com a condição de todos os eventos ocorrerem na mesma localização. Essa condição pode ser feita com o operador *inside*:

$$acumChuvaHMaior15Loc = \forall a(a.description = "intChuva") | a.value \geq 15 \wedge \quad (4.3)$$

$$\forall b(b.description = "intChuva" \wedge b.value \geq 15) |$$

$$b.space \text{ inside } a.space \wedge a.space \text{ inside } b.space[1hour]$$

$$chuvaForteLoc = \exists b | b.description = "acumChuvaD" \wedge b.value \geq 50 \quad (4.4)$$

$$\wedge \exists c | c.description = "acumChuvaHMaior15" \wedge$$

$$c.time \text{ during } b.time \wedge a.space \text{ inside } b.space$$

$$\wedge b.space \text{ inside } a.space$$

Dado esses padrões, os cenários 2 e 3 podem ser descrito da seguinte forma:

$$chuvaForteViz = \exists a | a.description = "chuvaForteLoc" \wedge \quad (4.5)$$

$$\exists b | b.description = "chuvaForteLoc" \wedge (a.time \text{ overlap } b.time \vee$$

$$a.time \text{ equal } b.time) \wedge a.space \text{ touch}[10km] b.space$$

$$chuvaForteRio = \exists a | a.description = "chuvaForteLoc" \wedge \quad (4.6)$$

$$\exists b | b.description = "rio" \wedge a.time \text{ during } b.time \wedge$$

$$b.space \text{ cross } a.space$$

O *Fluxo2* (Tabela 4.6) contém possíveis eventos resultantes do padrão *ChuvaForteLoc* (similares aos eventos gerados na seção anterior pelo padrão *chuvaForte*) e eventos extraídos do mapa hidrográfico do estado de São Paulo (*e500* e *e501*). As localizações dos eventos (no atributo *space*) correspondem aos objetos espaciais dispostos de acordo com a Figura 4.3.

A verificação do padrão *chuvaForteViz* sobre *Fluxo2* gera os eventos complexos *ec3* e *ec4*. O padrão *chuvaForteRio* detecta os eventos complexos *ec5* e *ec6*. A Figura 4.4 ilustra os eventos e as Tabelas 4.7 e 4.8 mostram os detalhes de suas estruturas.

Tabela 4.6: Eventos associados a *Fluxo2*, escondidos os atributos irrelevantes para o padrão

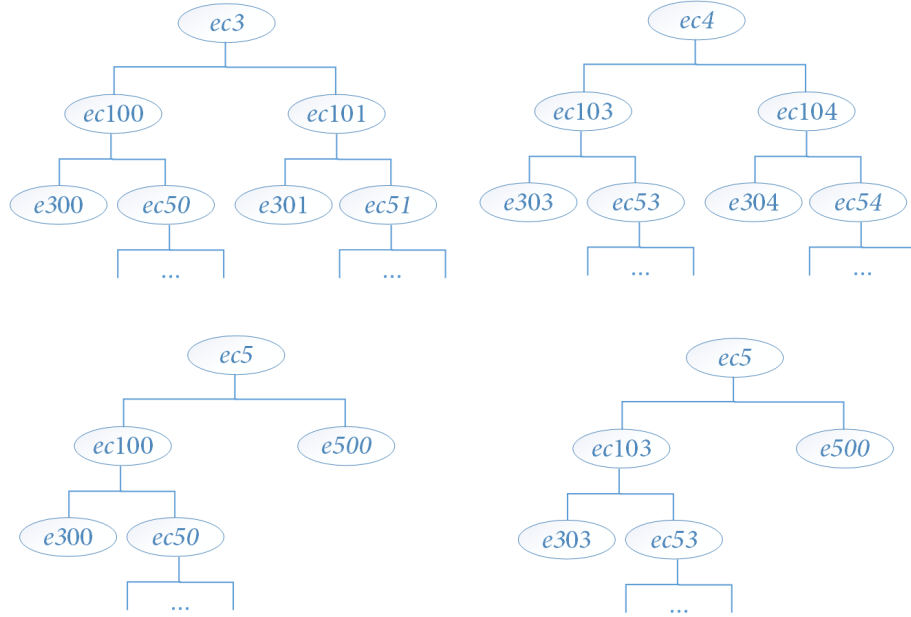
id	description	time		space	eventList
		startValidTime	endValidTime		
ec100	“chuvaForteLoc”	17:00 11/05/2015	18:00 11/05/2015	Campinas	e300, ec50
ec101	“chuvaForteLoc”	17:00 11/05/2015	18:00 11/05/2015	Hortolândia	e301, ec51
ec102	“chuvaForteLoc”	17:00 11/05/2015	18:00 11/05/2015	Aeroporto	e302, ec52
ec103	“chuvaForteLoc”	08:36 12/05/2015	09:36 12/05/2015	Campinas	e303, ec53
ec104	“chuvaForteLoc”	09:00 12/05/2015	10:00 12/05/2015	NÓdessa	e304, ec54
ec105	“chuvaForteLoc”	10:20 12/05/2015	11:20 12/05/2015	ECoelho	e305, ec55
e500	“rio”	00:00 01/03/2015	00:00 29/02/2016	Atibaia	-
e501	“rio”	00:00 01/03/2015	00:00 29/02/2016	Capivarí	-



Figura 4.3: Mapa de Campinas e Região.

Tabela 4.7: Estrutura dos eventos *ec3* a *ec6*, parte 1

id	description	time		space
		startValidTime	endValidTime	
ec3	“chuvaForteViz”	17:00 11/05/2015	18:00 11/05/2015	Campinas + Hortolândia
ec4	“chuvaForteViz”	09:36 12/05/2015	10:00 12/05/2015	Campinas + NÓdessa
ec5	“chuvaForteRio”	17:00 11/05/2015	18:00 11/05/2015	Campinas + Atibaia
ec6	“chuvaForteRio”	08:36 12/05/2015	09:36 12/05/2015	Campinas + Atibaia

Figura 4.4: Eventos complexos gerados pelos padrões *chuvaForteViz* e *chuvaForteLoc*.Tabela 4.8: Estrutura dos eventos *ec3* a *ec6*, parte 2

id	operatorList	eventList
ec3	<time, equal, time>, <space, touch, space>	ec100, ec101
ec4	<time, overlap, time>, <space, touch, space>	ec103, ec104
ec5	<time, during, time>, <space, cross, space>	ec100, e500
ec6	<time, during, time>, <space, cross, space>	ec103, e500

Discussão

Dados os cenários 2 e 3, o ponto forte da nossa proposta é a descrição de cenários que envolvam relações espaciais entre os eventos. A EPL da ferramenta Esper, ou as demais linguagens encontradas nesta pesquisa, não são capazes de descrever esses tipos de cenários. Além disso, os operadores espaciais podem ser aplicados considerando uma tolerância. Por exemplo, no cenário 2, o operador *touch* é usado com uma tolerância de $10km$ permitindo considerar não só regiões vizinhas, mas também regiões que distam menos de $2 \times 10km$ de Campinas. Isto permitiu que o evento *ec4* fosse gerado. Como mostrado pela Figura 4.3, os municípios de Nova Odessa e Campinas não são vizinhos mas se encaixam na tolerância especificada. Ainda assim, a abordagem espacial é limitada ao tratar relações entre múltiplos objetos por seus MBRs.

O ponto fraco da nossa abordagem nestes cenários é não adotar políticas de padrões, como políticas de cardinalidade e de repetição, apresentadas na seção 2.2. Essas políticas são definidas nas linguagens mais completas, como a EPL da Esper. No cenário 3, uma política de cardinalidade poderia decidir se outros eventos complexos devem ser criados para cada rio que cruza Campinas (gerando 4 eventos ao invés de 2) ou se basta considerar um rio (o rio Atibaia foi usado). A política de repetição poderia decidir qual evento (*e500* ou *e501*) participaria dos eventos *ec5* e *ec6*.

4.3 Cenário 4: Frente Fria em Campinas

De acordo com o projeto WW2010 da Universidade de Illinois [28], uma frente fria é definida como uma zona de transição em que uma massa de ar fria substitui uma massa de ar quente. A Tabela 4.9 apresenta as características ambientais desse evento, onde Ci, Cs, Cb e Cu são tipos de nuvens.

Tabela 4.9: Cenário antes, durante e após a passagem de uma frente fria, adaptado de Ahrens (1994) apud [28]

	antes de passar	enquanto passa	depois de passar
Vento	sul-sudoeste	tempestuoso, inconstante	oeste-noroeste
Temperatura	quente	queda súbita	queda constante
Pressão	queda constante	mínima, aumento brusco	aumento constante
Nuvens	aumentando: Ci, Cs e Cb	Cb	Cu
Precipitação	chuvas curtas	chuva forte	diminuindo
Visibilidade	razoável a ruim	ruim, depois melhorando	boa, exceto quando chove
Ponto de Orvalho	alto e estável	queda brusca	diminuindo

Esta seção mostra como especificar a presença de frente fria na cidade de Campinas, que ocorre quando as características da coluna “enquanto passa” da Tabela 4.9 são satisfeitas. Para simplificar o cenário, esta seção foca a descrição de três dos aspectos abordados: (a) temperatura, (b) vento e (c) precipitação.

A queda súbita de temperatura (aspecto (a)) é detectada pela diminuição da temperatura em x graus em um curto período. Para este cenário, supomos uma queda de no mínimo 3° em 2 horas. O padrão para detectar este cenário em Campinas é:

$$\begin{aligned} quedaTemp = \quad & \exists x \wedge \exists y [(x.description, y.description) = \text{“temp”} \wedge \\ & x.value \geq y.value + 3 \wedge x.time \text{ before } y.time \wedge \\ & x.space \text{ inside } y.space \wedge y.space \text{ inside } Campinas [2hours] \end{aligned} \quad (4.7)$$

O vento inconstante (aspecto (b)) pode ser constatado por medições com diferentes direções de vento em pontos distintos de uma mesma região no mesmo momento. Para esse cenário, supomos que a detecção de três diferentes direções de vento é suficiente. O padrão para detectar este cenário em Campinas é:

$$\begin{aligned} ventoInc = \quad & \exists x \wedge \exists y \wedge \exists z [(x.description, y.description, z.description) = \text{“dirVento”} \\ & \wedge \neq (x.value, y.value, z.value) \wedge equal(x.time, y.time, z.time) \wedge \\ & disjoint(x.space, y.space, z.space) \wedge \\ & (x.space, y.space, z.space) \text{ inside } Campinas \end{aligned} \quad (4.8)$$

A descrição de chuva forte (aspecto (c)) já foi apresentada pelas seções anteriores. Dessa forma, o padrão a seguir combina os aspectos (a), (b) e (c) para a detecção de frente fria em Campinas nas últimas 3 horas:

$$\begin{aligned} frentefriaCamp = \quad & quedaTemp \wedge ventoInc \wedge \exists x [x.description = \text{“chuvaFortLoc”} \\ & \wedge x.space \text{ inside } Campinas [3hours] \end{aligned} \quad (4.9)$$

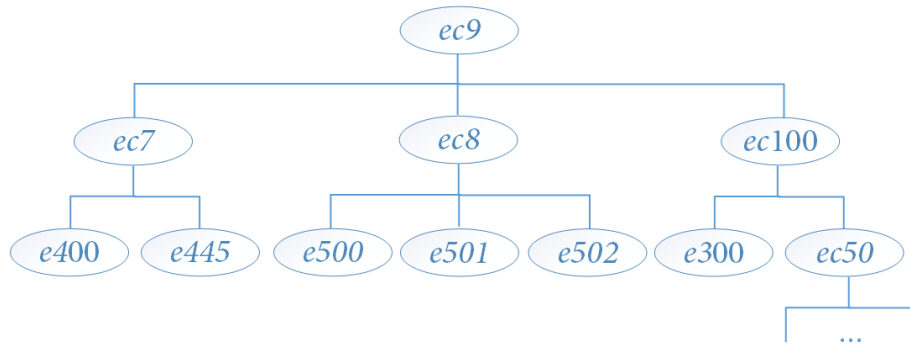
A verificação do padrão *frenteFriaCampinas* sobre o *Fluxo3* na Tabela 4.10 gera o evento complexo *ec9*, mostrado na Figura 4.5. O evento *ec9* engloba os eventos complexos intermediários *ec7*, *ec8* e *ec100* resultantes da verificação dos subpadrões *quedaTemp*, *ventoInc* e *chuvaForteLoc*, respectivamente. As Tabelas 4.11 e 4.12 mostram a estrutura dos eventos complexos. As disposições dos objetos espaciais são apresentadas na Figura 4.3 da seção anterior.

Discussão

Os cenários apresentados combinam todos os aspectos considerados na dissertação, evidenciando a amplitude de possibilidades de escrita de padrões. No cenário 4, além

Tabela 4.10: Eventos associados a *Fluxo3*

id	description	value	time		space	eventList
			startValidTime	endValidTime		
ec100	“chuvaFor” teLoc”	-	17:00 11/05/2015	18:00 11/05/2015	Campinas	e300 ec50
e400	“temp”	25	16:00 11/05/2015	16:02 11/05/2015	Aeroporto	-
e430	“temp”	23	17:00 11/05/2015	17:02 11/05/2015	Aeroporto	-
e445	“temp”	21	17:30 11/05/2015	17:32 11/05/2015	Aeroporto	-
e500	“dirVento”	“NE”	15:00 11/05/2015	15:02 11/05/2015	Aeroporto	-
e501	“dirVento”	“NO”	15:00 11/05/2015	15:02 11/05/2015	BGeraldto	-
e502	“dirVento”	“SO”	15:00 11/05/2015	15:02 11/05/2015	LPará	-

Figura 4.5: Evento complexo gerado pelo padrão *frenteFriaCamp*.Tabela 4.11: Estrutura dos eventos *ec7* a *ec9*, parte 1

id	description	time		space
		startValidTime	endValidTime	
ec7	“quedaTemp”	16:00 11/05/2015	17:32 11/05/2015	Aeroporto
ec8	“ventoInc”	15:00 11/05/2015	15:02 11/05/2015	Aeroporto+BGeraldto+LPará
ec9	“frenteFriaCamp”	15:00 11/05/2015	17:32 11/05/2015	Campinas

Tabela 4.12: Estrutura dos eventos *ec7* a *ec9*, parte 2

id	operatorList	eventList
ec7	<description, =, “temp”>, <value, ≥, value+13>, <time, before, time>, <space, inside, space>, <space, inside, campinas> <time, before, time>, <space, inside, space>	e400, e445
ec8	<description, =, “dirVento”>, <value, ≠, value> <time, equal, time>, <space, disjoint, Campinas>, <space, inside, Campinas>	e500, e501, e502
ec9	<time, within, 3 hours>	ec7, ec8, ec100

de reusar um evento complexo gerado por outro padrão (*chuvaForteLoc*), a construção hierárquica do padrão *frenteFriaCamp* reusa os padrões *quedaTemp* e *ventoInc*. O reuso de padrões visa a simplificação da escrita de cenários e facilita a colaboração entre cientistas. Por outro lado, o reuso de eventos complexos visa a construção de cenários ainda mais elaborados, relacionando eventos complexos a outros eventos.

Além disso, o uso de operadores de forma agregada evita a repetição de predicados, principalmente no padrão *ventoInc* em que todos os operadores são usados dessa forma. Como citado na seção 2.2.3, Li et al. [18] implementam um operador chamado ISEQ para agregar diferentes relações temporais em sequência. Sua agregação simplifica o uso repetitivo das relações temporais de Allen [4], utilizando apenas combinações dos operadores numéricos ($=$, $<$, $<=$, $>$ e $>=$) sobre os atributos de início e fim de um intervalo temporal. Na nossa linguagem, essa simplificação é semelhante à aplicação dos operadores *equal* e *before* sobre os atributos *startValidTime* e *endValidTime*. Nossa proposta de agregação permite agregar outros tipos de relações (sejam elas numérica, literal, temporal ou espacial), adotando a semântica sequencial e, em alguns casos, total. Neste caso, diferentes usos de uma mesma relação são agregados, aplicando um único operador a uma lista de atributos a qual a relação se aplica.

Essa abordagem permite a simplificação da escrita de padrões e evidencia os relacionamentos em comum entre eventos. Inclusive, dada esta notação, pode se conceber implementações de vários tipos. Por exemplo, um padrão com a seguinte aplicação tradicional de operadores *a.time before b.time \wedge b.time before c.time*, é implementado com geração de 2 eventos complexos: o primeiro relacionando *a* e *b* e o segundo *b* e *c*. No entanto, um padrão escrito com a seguinte forma agregada *before(a.time, b.time, c.time)* tem a mesma semântica do padrão anterior (uma vez que sua leitura é sequencial) e garante o relacionamento entre os três eventos, podendo eventualmente ser implementada via um único evento complexo a partir desta relação.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

A principal contribuição desta dissertação é uma linguagem para especificar e detectar cenários de interesse a partir de variáveis ambientais, ajudando pesquisadores em biodiversidade na análise de fenômenos e correlação de dados. O trabalho utiliza a tecnologia CEP, que permite a detecção de cenários em tempo real, onde os dados ambientais são tratados como eventos e os cenários são descritos por padrões de eventos de acordo com a linguagem proposta.

Este trabalho estende a tese de Koga [17], adicionando a detecção de padrões mais complexos ao *framework* que integra dados heterogêneos e detecta padrões simples. Em particular, a linguagem proposta apresenta as seguintes características originais de CEP:

- suporte a dados em múltiplas escalas temporais e espaciais;
- uso de operadores de forma agregada/compacta;
- suporte a operadores temporais e espaciais;
- composição hierárquica de padrões e eventos complexos.

Propostas na literatura de CEP não consideram relacionamentos espaciais entre eventos e, ainda, são limitadas na geração de eventos complexos resultantes da detecção de um padrão. Na linguagem proposta, toda detecção de cenário gera eventos complexos de forma hierárquica, que podem participar de outras construções.

5.2 Trabalhos Futuros

Direções que podem ser exploradas a partir deste trabalho são:

- **Implementação da linguagem proposta** - O trabalho especifica a sintaxe e semântica da linguagem proposta, mas não a implementa;
- **Extensão da linguagem com outros elementos** - A linguagem foca na descrição e padrões baseados na especificação de eventos de interesse e suas relações. Portanto, não define políticas de padrões (apresentadas na seção 2.2) ou operações para, por exemplo, manipulação de fluxo de dados ou para controle de saída dos padrões, que geralmente são incluídas em uma EPL completa (detalhes em 3.4). Por exemplo, a EPL da ferramenta Esper [35] implementa esses e outros elementos, apesar de não suportar relacionamentos espaciais e ser limitada ao tratar da geração de eventos complexos;
- **Algoritmo para detecção e composição de eventos complexos** - Desenvolver um algoritmo que gere eventos complexos de forma hierárquica durante a detecção de um padrão descrito pela linguagem proposta nesta dissertação. Por exemplo, Xie e Taylor [38] desenvolvem um algoritmo de *backtracking* que, a partir do padrão, gera uma árvore de análise (*parse tree*) e condiciona a verificação do padrão a essa árvore. Assim, os eventos são casados de acordo com a árvore de análise, partindo das folhas até a raiz, gerando os eventos complexos correspondentes aos nós não-folha. No entanto, esse mecanismo de detecção e geração de eventos complexos é limitada à uma linguagem simples que suporta poucos operadores;
- **Algoritmo para detecção e composição de eventos complexos visando otimização** - Uma outra extensão seria o estudo de otimizações em tempo de execução (como em [2, 37]), no custo de transmissão em fontes distribuídas (como em [3]) e entre outras otimizações;
- **Detecção de cenários com o apoio de Aprendizado de Máquina** - A especificação de cenários de interesse não é uma tarefa trivial. Alguns cenários ainda estão sendo estudados por especialistas, não tendo um modelo pré-definido; outros cenários se apresentam em diferentes formas, ou ainda são demasiadamente complexos de modelar. Dessa forma, técnicas de aprendizado de máquina podem ajudar a especificação de um cenário que, posteriormente, seja detectado usando CEP. Exemplos de propostas nesse sentido são: Eliades et al. [10], no entendimento da reação do cloro e detecção de contaminação na água potável; e Li et al. [19], na detecção de vários tipos de tempestades usando técnicas de Mineração de Dados;
- **Interface para compor padrões de forma mais intuitiva** - A linguagem proposta não é amigável e é necessário desenvolver uma interface que ajude especialistas a especificar seus cenários. Um exemplo de proposta nesse sentido é Sen et al. [33];

- **Definição de janelas de aplicação de regras** - Havendo muitas regras, pode haver uma explosão de análise de padrões. Uma extensão seria definir janelas para detecção de eventos, visando otimizar o desempenho.

Referências Bibliográficas

- [1] Charu C. Aggarwal and Karthik Subbian. Event detection in social streams. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 624–635.
- [2] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. Efficient pattern matching over event streams. In *Proceedings of the ACM SIGMOD*, pages 147–160, 2008.
- [3] Mert Akdere, Uğur Çetintemel, and Nesime Tatbul. Plan-based complex event detection across distributed sources. *Proceedings of the VLDB Endowment*, 1(1):66–77, August 2008.
- [4] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.
- [5] Roger S. Barga and Hillary Caituiro-Monge. Event correlation and pattern detection in cedr. In *Proceedings of the EBDT*, pages 919–930, 2006.
- [6] Central Wheeler Bureau. Heavy rain. <http://www.cwb.gov.tw/V7e/observe/rainfall/define.htm>. Accessed: 2015-05-01.
- [7] Gilberto Câmara, Marco Casanova, Andrea Hemerly, Geovane Magalhães, and Claudia Medeiros. *Anatomia de sistemas de informação geográfica*. INPE, 1996.
- [8] Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *Proceedings of the Third International Symposium on Advances in Spatial Databases, SSD '93*, pages 277–295, London, UK, UK, 1993. Springer-Verlag.
- [9] Jürgen Dunkel. On complex event processing for sensor networks. In *Proceedings of the International Symposium on Autonomous Decentralized Systems, 2009. ISADS '09.*, pages 1–6, March 2009.

- [10] Demetrios G. Eliades, Christos Panayiotou, and Marios M. Polycarpou. Contamination event detection in drinking water systems using a real-time learning approach. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 663–670, July 2014.
- [11] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co, 2010.
- [12] Glaucia Faria. A spatio-temporal database for development of applications in geographic information systems. Master’s thesis, Instituto de Computação - Unicamp, 1998.
- [13] Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399, October 1994.
- [14] Alex Hardisty and Dave Roberts. A decadal view of biodiversity informatics: challenges and priorities. *BMC Ecology*, 13(1), 2013.
- [15] Mitchell C Kerman, Wei Jiang, Alan F Blumberg, and Samuel E Buttrey. Event detection challenges, methods, and applications in natural and artificial systems. In *Proceedings of the 14th International Command and Control Research and Technology Symposium, ICCRTS ’09*, pages 1–19, Washington, DC, USA, 2009. CCRP.
- [16] Mitchell C. Kerman, Wey Jiang, Alan F. Blumberg, and Samuel E. Buttrey. The application of a quantile regression metamodel for salinity event detection confirmation within new york harbour oceanographic data. *Journal of Operational Oceanography*, 2(1):49–70, 2009.
- [17] Ivo Kenji Koga. *An Event-Based Approach to Process Environmental Data*. PhD thesis, Instituto de Computação - Unicamp, September 2013. Supervisor Claudia Bauzer Medeiros.
- [18] Ming Li, Murali Mani, Elke A. Rundensteiner, and Tao Lin. Complex event pattern detection over streams with interval-based temporal semantics. In *Proceedings of the 5th ACM international conference on Distributed event-based system, DEBS ’11*, pages 291–302, New York, NY, USA, 2011. ACM.
- [19] Xiang Li, Beth Plale, Nithya Vijayakumar, Rahul Ramachandran, Sara Graves, and Helen Conover. Real-time storm detection and weather forecast activation through data mining and events processing. *Earth Science Informatics*, 1(2):49–57, 2008.

- [20] Mo Liu, Elke Rundensteiner, Kara Greenfield, Chetan Gupta, Song Wang, Ismail Ari, and Abhay Mehta. E-cube: Multi-dimensional event sequence analysis using hierarchical pattern query sharing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 889–900, New York, NY, USA, 2011. ACM.
- [21] Alejandro Llaves and Chris Renschler. Observing changes in real-time sensor observations. In *Proceedings of the 15nd International Conference on Geographic Information Science*, AGILE'2012, pages 388–392, Avignon, France, April 2012.
- [22] David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [23] Amina Madani, Omar Boussaid, and Djamel Eddine Zegour. What's happening: A survey of tweets event detection. In *Proceedings of the Third International Conference on Communications, Computation, Networks and Technologies*, INNOV 2014, pages 16–22, Nice, France, October 2014.
- [24] Aziz Nasridinov, Sun-Young Ihm, Young-Sik Jeong, and Young-Ho Park. Event detection in wireless sensor networks: Survey and challenges. In James J. (Jong Hyuk) Park, Hojjat Adeli, Namje Park, and Isaac Woungang, editors, *Mobile, Ubiquitous, and Intelligent Computing*, volume 274 of *Lecture Notes in Electrical Engineering*, pages 585–590. Springer Berlin Heidelberg, 2014.
- [25] Navscales. Navscales: Navigating through scales in space, time and knowledge domains. <http://www.lis.ic.unicamp.br/projects/navscales-navigating-through-scales-in-space-time-and-knowledge-domains/>. Accessed: 2013-10-03.
- [26] Arif Nurwidyanoro and Edi Winarko. Event detection in social media: A survey. In *Proceedings of the International Conference on ICT for Smart Society (ICISS)*, pages 1–5, June 2013.
- [27] Hannes Obweiger, Josef Schiefer, Peter Kepplinger, and Martin Suntinger. Discovering hierarchical patterns in event-based systems. In *Proceedings of the SCC*, pages 329–336, 2010.
- [28] University of Illinois. Cold front. [http://ww2010.atmos.uiuc.edu/\(Gh\)/home.rxml](http://ww2010.atmos.uiuc.edu/(Gh)/home.rxml). Accessed: 2015-05-01.
- [29] Nikos Pelekis, Babis Theodoulidis, Ioannis Kopanakis, and Yannis Theodoridis. Literature review of spatio-temporal database models. *Knowl. Eng. Rev.*, 19(3):235–274, September 2004.

- [30] Peter R. Pietzuch, Brian Shand, and Jean Bacon. Composite event detection as a generic middleware extension. *IEEE Network*, 18(1):44–55, 2004.
- [31] Julien Sauvageon, Alice M. Agogino, Ali F. Mehr, and Irem Y. Tumer. Comparison of event detection methods for centralized sensor networks. In *Proceedings of the 2006 IEEE Sensors Applications Symposium*, pages 95–100, 2006.
- [32] Sinan Sen and Nenad Stojanovic. Gruve: a methodology for complex event pattern life cycle management. In *Proceedings of the 22nd International Conference on Advanced information systems engineering*, CAiSE’10, pages 209–223, 2010.
- [33] Sinan Sen, Nenad Stojanovic, and Ruofeng Lin. A graphical editor for complex event pattern generation. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS ’09, pages 41:1–41:2, New York, NY, USA, 2009. ACM.
- [34] Sinan Sen, Nenad Stojanovic, and Ljiljana Stojanovic. An approach for iterative event pattern recommendation. In *Proceedings of the DEBS*, pages 196–205, 2010.
- [35] Esper Team and EsperTech Inc. Esper reference. <http://www.espertech.com/esper/>, 2015. Accessed: 2015-02-10.
- [36] Colorado State University. Colorado water watch: Anomaly detection methodology. <http://waterwatch.colostate.edu/Home/Learn>, 2013. Acessado: 2015-05-01.
- [37] Louis Woods, Jens Teubner, and Gustavo Alonso. Complex event detection at wire speed with fpgas. *Proceedings of the VLDB Endow.*, 3(1-2):660–669, sep 2010.
- [38] Ping Xie and David Taylor. Specifying and locating hierarchical patterns in event data. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative research*, CASCON ’04, pages 81–95, 2004.
- [39] Michael Zoumboulakis and George Roussos. Escalation: Complex event detection in wireless sensor networks. In Gerd Kortuem, Joe Finney, Rodger Lea, and Vasughi Sundramoorthy, editors, *Smart Sensing and Context*, volume 4793 of *Lecture Notes in Computer Science*, pages 270–285. Springer Berlin Heidelberg, 2007.