3 Uma Ontologia para Sistemas de Gerência de Análises em Biossequências

3.1 Introdução

Este capítulo apresenta uma ontologia que direciona o sistema de gerência de análises em biossequências. Serão tratados aqui os aspectos gerais, comuns aos ambientes de trabalho dos pesquisadores em que este sistema pode ser implementado. Os aspectos da ontologia particulares aos ambientes de trabalho dos pesquisadores serão apresentados, e justificados, ao longo dos capítulos seguintes. Eles estão omitidos aqui para evitar uma discussão prematura sobre estes ambientes.

A ontologia possui descrições das classes relevantes ao domínio de análise em biossequências, suas propriedades e como elas se relacionam. As principais classes da ontologia referem-se a processos, recursos, dados e projetos comumente envolvidos em análises de biossequências, conforme resumido esquematicamente na Figura 3. A ontologia é flexível, ou seja, permite acrescentar, atualizar e remover classes, propriedades e instâncias.

A ontologia possui ainda metadados para todas as classes, propriedades e instâncias consideradas. Os metadados são os conceitos básicos definidos pelo padrão Dublin Core: pesquisador que criou a classe, propriedade ou instância; data de criação; descrição dada pelo pesquisador.

A descrição da ontologia, apresentada neste capítulo, não se compromete com nenhuma linguagem. Porém, utiliza conceitos básicos de RDF e RDF Schema [W3C, 2004a] como classes, propriedades e instâncias. Na implementação do sistema de gerência de análises em biossequências, a ontologia foi escrita em Amzi-Prolog [Amzi!, 2004], que possui uma sintaxe simples e oferece uma API para Java, linguagem utilizada para implementar o sistema. Por razões semelhantes, a implementação poderia ter utilizado a linguagem OWL [W3C, 2003a, b] e o *framework* JENA. O anexo apresenta a descrição da ontologia em Amzi-Prolog e em OWL.

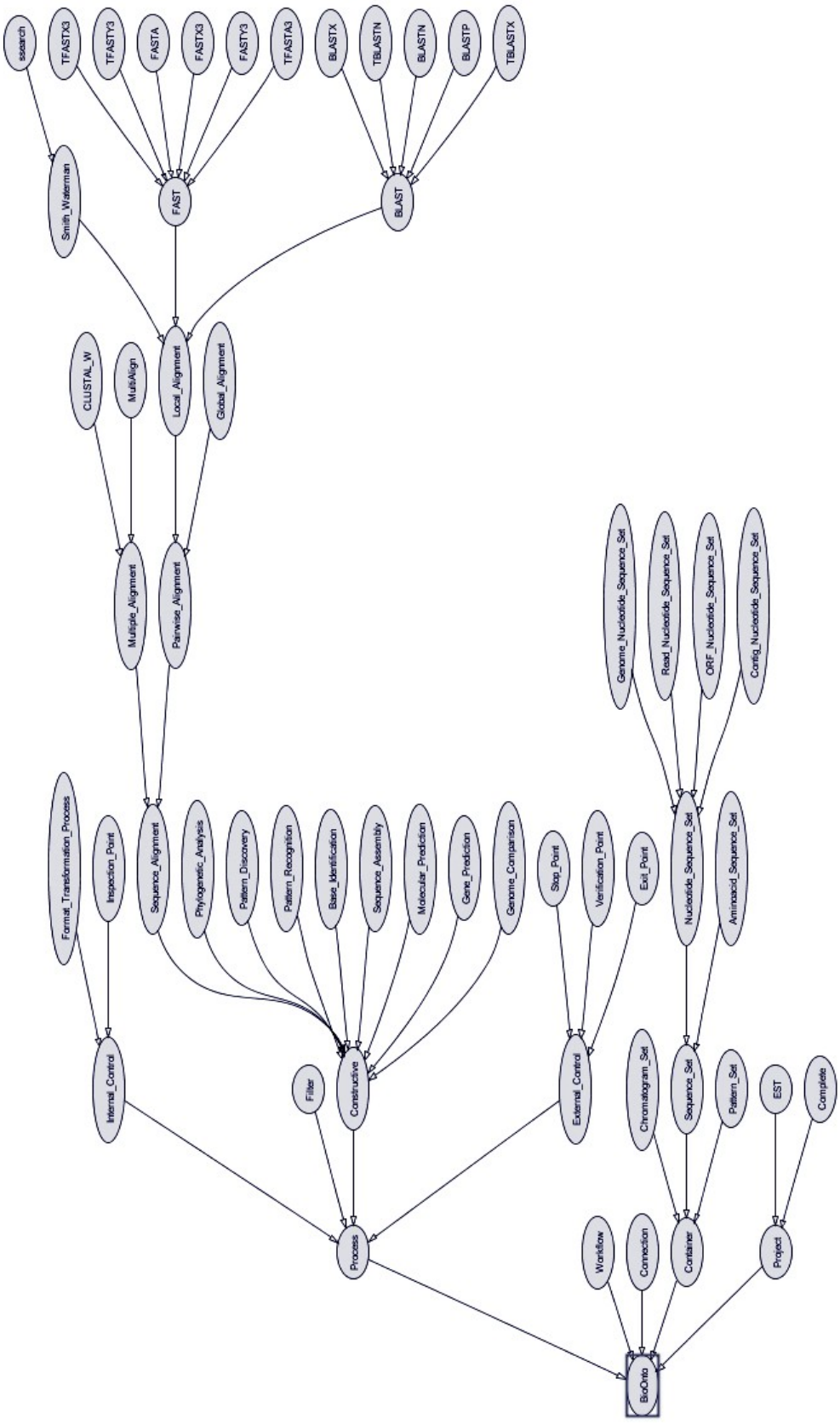

Figura 3. Diagrama de classes da ontologia.

3.2 Workflow

A classe Workflow possui instâncias, chamadas de *workflows de Bioinformática* ou, simplesmente, *workflows*.

Um workflow define uma composição de *processos,* modelando chamadas a programas de Bioinformática que analisam biossequências e que ajudam um pesquisador a interpretá-las. Desta forma, um workflow pode conter vários processos que modelam chamadas distintas ao mesmo programa de Bioinformática, cada uma possivelmente especificando valores diferentes para os parâmetros do programa.

A transferência de dados de um processo para outro é intermediada por *contêineres.* Assim, um processo que *produz* dados possui uma *conexão chegando* em um contêiner, e um processo que *consome* dados possui uma *conexão saindo* de um contêiner.

Processos, conexões e contêineres são instâncias das classes Processo, Conexão e Contêiner, definidas nas seções 3.3, 3.4e 3.5, respectivamente.

A estrutura de um workflow é modelada definindo-se três propriedades para a classe Workflow (os nomes das propriedades foram escolhidos para não confundir com os nomes das classes):

- Processo-utilizado, que relaciona um workflow a um processo;
- Contêiner-utilizado, que relaciona um workflow a um contêiner;
- Conexão-utilizada, que relaciona um workflow a uma conexão; e
- Projeto-associado, que relaciona um workflow a um ou a vários projetos.

Todas estas propriedades são multi-valoradas já que, naturalmente, um workflow pode ter mais de um processo, contêiner ou conexão.

Pela própria noção de conexão, um workflow induz um grafo bipartido, onde há nós representando processos, chamados de *nós-processo*, nós representando contêineres, chamados *nós-contêiner*, e arcos que conectam nós-processo e nós-contêiner, chamados de *arcos de conexão.*

A Figura 4 apresenta um exemplo onde o nó rotulado por p_p representa um nó-processo produtor, o nó rotulado por p_c representa um nó-processo consumidor e c_1 representa o nó-contêiner que armazena os dados compartilhados por p_p e p_c.

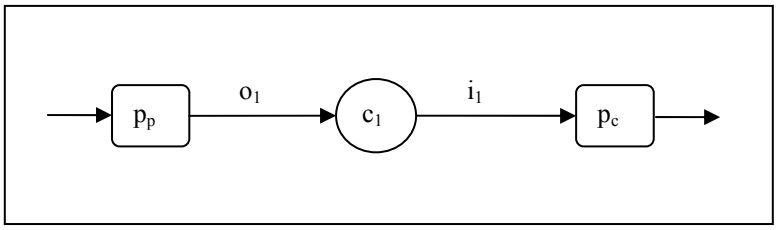

Figura 4. Representação de workflow por grafo bipartido.

3.3 Processos

3.3.1 Classes de Processos

A classe Processo possui instâncias, chamadas de *processos*, que modelam chamadas aos programas de Bioinformática. Com já foi salientado, um workflow pode conter vários processos que modelam chamadas ao mesmo programa de Bioinformática, cada uma possivelmente especificando valores diferentes para os parâmetros do programa.

A classe Processo possui as seguintes sub-classes:

- Construtivo;
- Filtro;
- Controle Interno; e
- Controle Externo.

Uma instância da classe Construtivo é chamada de *processo construtivo*, e assim por diante para as outras sub-classes de Processo.

Intuitivamente, um *processo construtivo* cria novos conjuntos de dados, pertinentes ao domínio de análise de Bioinformática. Através de discussões com biólogos, de um levantamento elaborado de acordo com questionários respondidos por pesquisadores em [Stevens et al., 2004] e pesquisa bibliográfica exaustiva, identificou-se quais são as principais tarefas dos biólogos e, consequentemente, quais são as classes de processos construtivos que um sistema de gerência de workflows de Bioinformática deve contemplar. O Quadro 1 apresenta as sub-classes da classe Construtivo, que correspondem às tarefas de nomeação de bases, montagem de fragmentos, alinhamento de sequências, comparação de genomas, predição de estruturas terciárias e secundárias, análise filogenética, descoberta de

padrões, reconhecimento de padrões, predição de genes e outras atividades que não se encaixam em nenhuma das definidas anteriormente .

- Bases Identification
- Sequence Assembly
- Sequence Alignment
- Genome Comparison
- Molecular Prediction
- Phylogenetic Analysis
- Pattern Discovery
- Pattern Recognition
- Gene Prediction
- Miscellany

Quadro 1. Sub-classes dos processos construtivos.

O esquema do Quadro 2 apresenta a hierarquia das sub-classes dos processos de alinhamento de sequência, terminando em classes cujas instâncias são os processos que de fato comporão um workflow. Por exemplo, as instâncias das classes BLASTP e FASTY3 modelam chamadas aos programas BLASTP e FASTY, com seus parâmetros específicos.

Um *processo filtro* analisa um conjunto de dados gerado por um processo e extrai partes dele para futuro processamento.

Um *processo de controle interno* é um processo acrescentado automaticamente pelo sistema de gerência de workflows para que o funcionamento do workflow se torne coerente, viável ou mais eficiente. Esta classe de processos engloba os processos de transformação de formato e os processos de inspeção, entre outros.

Um *processo de transformação de formato,* como o nome indica, aplica uma transformação de formato em um conjunto de dados. Uma transformação de formato pode ser *simples*, como uma mera transformação sintática, ou *elaborada*, como a transformação de uma sequência de nucleotídeos em uma sequência de aminoácidos.

- Sequence Alignment
  - Multiple Alignment
    - CLUSTAL W
    - MultiAlign
  - Pairwise Alignment
    - Global Alignment
      - Needleman
    - Local Alignment
      - Smith Waterman
        - ssearch
      - FAST
        - FASTA
        - FASTY3
        - FASTX3
        - TFASTA3
        - TFASTX3
        - TFASTY3
      - BLAST
        - BLASTP
        - BLASTN
        - BLASTX
        - TBLASTX
        - TBLASTN

Quadro 2. Sub-classes dos processos de alinhamento de sequências.

Um *processo de inspeção* verifica se os dados de entrada e o resultado da execução de um processo estão corretos ou não. Sendo conhecidos os formatos da saída gerada e da entrada esperada por um determinado processo, caso não estejam de acordo com o esperado, a inspeção indicará um erro.

Outros processos internos serão introduzidos, e justificados, ao longo dos capítulos seguintes. Eles estão omitidos aqui para evitar uma discussão prematura sobre detalhes de otimização e validação, entre outros.

Como exemplo, temos:

- união de contêineres (∪): usada para criar um contêiner pela união de dois ou mais contêineres (entendidos como conjuntos de dados);

- armazenamento de dados: usado para armazenar dados gerados pelos workflows no *data warehouse* do SGABio.

Um processo construtivo pode estar relacionado a processos de inspeção, filtro ou transformação de formato. Por exemplo, considere um processo construtivo representando uma chamada ao BLASTP. É possível construir um processo de inspeção que analise os seus resultados e indique se a chamada ao BLASTP terminou ou não sua execução com sucesso. Além disso, um processo filtro pode relatar somente os alinhamentos resultantes do BLASTP cuja pontuação seja maior que um determinado valor, e um processo de transformação de formato pode modificar a saída do formato original do BLASTP para o formato XML [W3C, 2004c], necessário como entrada para outra instância de processo.

Um *processo de controle externo* ajuda o pesquisador a gerenciar a execução do workflow. A ontologia contempla os seguintes processos de controle externo:

- ponto de parada (!): indica um ponto em que a execução do workflow deve parar temporariamente para que o pesquisador analise os resultados intermediários já gerados;
- ponto de saída (*exit*): indica um ponto em que a execução do workflow deve parar; e
- ponto de verificação (*if*): usado sempre em conjunto com um processo *p*, indica uma condição, avaliada sobre os contêineres de entrada de *p*, que deve ser satisfeita para que *p* seja executado.

Os dois primeiros processos de controle externo podem ser tratados como qualquer outro processo do grafo bipartido da Figura 4. Entretanto, note que um ponto de verificação p_v para um processo p_c deve analisar os dados armazenados nos contêineres de entrada de p_c antes de p_c executar (ou não). Desta forma, um ponto de verificação p_v será acrescido ao workflow como se fizesse parte do processo p_c.

Assim, para representar pontos de verificação, é necessário uma pequena modificação no grafo bipartido que representa um workflow, como indicado esquematicamente na Figura 5. Com esta modificação, tanto o ponto de verificação p_v quanto o processo p_c lerão dados do contêiner c_1. A diferença é que, dependendo dos dados armazenados em c_1, o processo consumidor p_c não será selecionado para execução.

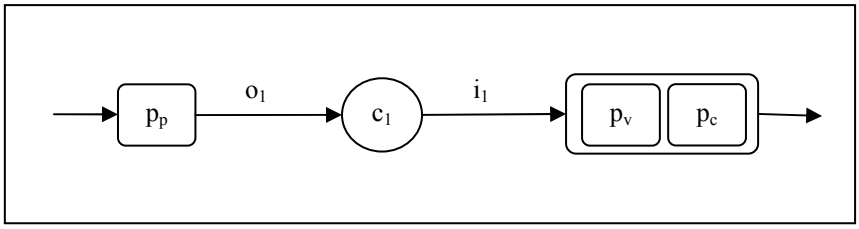

Figura 5. Processo de controle externo de verificação.

A ontologia pode ainda contemplar outros tipos de processos de controle externo como, por exemplo, a iteração (*while*), que indica uma lista de instâncias de processos que serão executados várias vezes até que uma condição de parada seja satisfeita. Como o ponto de parada de uma iteração depende dos dados gerados por um processo produtor, seria necessário fazer adaptações no grafo bipartido apresentado anteriormente. Por exemplo, teria que ser possível atribuir um valor inicial ao contêiner que armazenará os dados que serão analisados, e incluir um processo de iteração para analisar os dados que forem sendo armazenados em tal contêiner.

3.3.2 Propriedades de Processos

A classe Processo possui propriedades, classificadas em *parâmetros* e *propriedades de qualidade*.

3.3.2.1 Parâmetros

O comportamento de um programa de Bioinformática pode ser modificado através da configuração de seus parâmetros, que possuem um valor *default*.

Como grande parte dos programas são baseados em heurísticas e, consequentemente, não geram um resultado ótimo, existem parâmetros que restringem ou afrouxam a qualidade do resultado gerado, ou seja, a frequência de resultados falsos positivos (resultados falsos encontrados) e de falsos negativos (resultados verdadeiros não encontrados) pode variar.

Por exemplo, os processos da classe BLAST possuem como parâmetro *e-value*, que pode ser alterado para restringir a frequência de similaridades falsas encontradas e de similaridades verdadeiras não encontradas. Quanto menor o e-value, mais significativo é o alinhamento obtido.

Da mesma forma, os processos das classes Glimmer, Transterm e tRNAScan possuem parâmetros que restringem ou afrouxam a predição de genes, terminadores e tRNAs, respectivamente.

Existem outros parâmetros que possuem o objetivo de configurar o formato dos resultados dos processos. Neste caso, não é necessário a utilização de um processo de controle interno de transformação de formato.

3.3.2.2 Propriedades de Qualidade

As propriedades de qualidade são, na verdade, propriedades para as diversas sub-classes de Processo (e não para as suas instâncias), possuindo valores padrão.

A ontologia proposta define as seguintes meta-propriedades de qualidade:

- desempenho: medida da quantidade de recursos computacionais (como tempo de CPU, acesso a disco, etc.) consumidos pelos processos de uma classe;
- popularidade: medida da porcentagem de pesquisadores que conhecem e utilizam os processos de uma classe;
- custo: medida do custo financeiro para execução dos processos de uma classe;
- padrão: medida que indica o quanto os processos de uma classe são indicados como opção padrão para o tipo de tarefa a que se propõem;
- fidelidade: medida de quão próximo do ótimo estão, normalmente, os resultados gerados pelos processos de uma classe. Como a maioria das análises feitas pelos pesquisadores de Bioinformática são baseadas em heurísticas, os resultados podem conter erros, (falsos positivos e falsos negativos) e por isso é muito importante a qualidade fidelidade, pois ela informa ao pesquisador quais processos de uma mesma classe geram resultados mais confiáveis que outros.
- adequação: indica se os processos de uma classe são ou não adequados para um determinado tipo de projeto.

As propriedades de qualidade são definidas para diferenciar as classes de processos. Baseado nas qualidades que o pesquisador priorizar, o sistema de gerência de workflows pode ajudar o pesquisador a decidir qual é o processo mais

adequado em cada caso. Por exemplo, suponha que o pesquisador queira calcular um alinhamento local entre uma sequência nova e um determinado banco de dados. Esta tarefa pode ser feita por processos das classes BLAST e FAST, dependendo das propriedades de qualidade que o pesquisador definir. Se o pesquisador preferir desempenho, o sistema indicará o BLAST, mas se ele preferir fidelidade, o sistema escolherá a classe FAST.

Os valores das propriedades de qualidade para uma classe de processos podem modificar-se com o tempo. Por exemplo, o sistema pode atualizar automaticamente a ontologia quando descobrir novos fatos. Consequentemente, se o sistema perceber que a frequência de escolha de um processo aumentou, a propriedade popularidade pode ser atualizada.

Outras qualidades serão introduzidas ao longo dos capítulos seguintes. Elas estão omitidas aqui para evitar uma discussão prematura sobre detalhes de ambientes de trabalho dos pesquisadores onde o SGABio será implementado.

3.4 Conexões

A classe Conexão possui instâncias, chamadas *conexões*, que modelam as conexões ligando processos a contêineres. Possui as seguintes propriedades:

- Tipo, que pode assumir os valores ‘gradativo’ e ‘não-gradativo’;
- Origem, que assume valores da classe Contêiner ou da classe Processo;
- Destino, que assume valores da classe Processo, se o valor de Origem para a instância for da classe Contêiner, ou da classe Contêiner, se o valor de Origem para a instância for da classe Processo.

Uma conexão é *de leitura* (ou *de escrita*) quando a propriedade Origem para a instância for da classe Contêiner (ou da classe Processo). Intuitivamente, uma conexão de leitura indica que o processo lê itens de dados do contêiner; neste caso, o processo é chamado de *consumidor*. Simetricamente, uma conexão de escrita indica que o processo escreve itens de dados no contêiner; neste caso o processo é chamado de processo *produtor*.

Uma conexão é *gradativa* (ou *não-gradativa*) quando a propriedade Tipo assume o valor ‘gradativo’ (ou ‘não-gradativo’). Intuitivamente, esta propriedade indica como o processo escreve os itens de dados no contêiner, ou lê itens de dados do contêiner de acordo com a descrição da Tabela 1 e da Tabela 2.

Como exemplo, considere o processo Phrap, que realiza montagem de fragmentos. O Phrap lê um conjunto de *reads* (sequências de nucleotídeos) como dados de entrada e gera um conjunto de contigs (sequências de nucleotídeos) como dados de saída. Para realizar a montagem de fragmentos, o Phrap precisa analisar todos os *reads*, o que implica dizer que o Phrap lê os *reads* de forma não-gradativa. Durante a análise, os contigs gerados pelo Phrap são gerados um a um, o que significa dizer que o Phrap gera os contigs de forma gradativa.

Tabela 1 – Conexão de leitura gradativa e não-gradativa.

Conexão de leitura	Definição
gradativa	consumidor pode ler item de dados assim que for liberado para leitura consumidor lê cada item de dados uma vez, e apenas uma vez
não-gradativa	consumidor começa a ler apenas depois de todos os itens de dados estarem liberados para leitura consumidor pode reler item de dados

Tabela 2 – Conexão de escrita gradativa e não-gradativa.

Conexão de escrita	Definição
gradativa	produtor libera item de dados para leitura imediatamente após escrevê-lo produtor escreve item de dados uma vez, e apenas uma vez
não-gradativa	produtor libera os itens de dados para leitura apenas depois de escrever todos os itens de dados produtor pode rescrever item de dados

3.5 Contêineres

3.5.1 Classes de Contêineres

A classe Contêiner possui instâncias, chamadas *contêineres*, que modelam as estruturas de dados responsáveis por armazenar e gerenciar os conjuntos de dados que são compartilhados no workflow. Os contêineres que não são destino de alguma conexão e que modelam bancos de dados são também chamados de *recursos*.

O estudo das principais tarefas feitas pelos pesquisadores em Bioinformática permitiu não somente a definição de hierarquias de classes de processo, mas também a definição de hierarquias de classes de contêineres. A identificação destas hierarquias facilita a construção de sistemas que auxiliem pesquisadores a decidir quais processos são mais adequados em cada caso.

O Quadro 3 apresenta as sub-classes da classe Contêiner mais freqüentemente observadas. Já o Quadro 4 mostra algumas instâncias de recursos, em negrito, abaixo das classes a que pertencem.

- Chromatogram Set
- Sequence Set
  - Nucleotide Sequence Set
    - Contig Nucleotide Sequence Set
    - Read Nucleotide Sequence Set
    - ORF Nucleotide Sequence Set
    - Genome Nucleotide Sequence Set
  - Aminoacid Sequence Set
- Pattern Set

Quadro 3. Sub-classes de contêineres.

- Sequence Set
  - Nucleotide Sequence Set
    - **Genbank-NT**
    - **EMBL**
  - Aminoacid Sequence Set
    - **Genbank-NR**
    - **PIR**
    - **Swiss-Prot**
    - **Tr-EMBL**
- Pattern Set
  - **Prosite**
  - **PFam**
  - **Blocks**

Quadro 4. Instâncias de recursos.

3.5.2 Propriedades de Contêineres

3.5.2.1 Parâmetros

Um contêiner possui as seguintes propriedades:

- Tipo, que pode assumir os valores: ‘gradativo’, ‘não-gradativo’, ‘misto’;
- Tamanho Mínimo Estimado, que assume valores inteiros;
- Tamanho Máximo Estimado, que assume valores inteiros;
- Tipo de Acesso, que assume os valores ‘privado’ ou ‘público’;
- Formato de Dados, que assume valores tirados de um conjunto de *formatos*.

Um contêiner é *gradativo, não-gradativo* ou *misto,* dependendo do valor da propriedade Tipo. Esta propriedade dos contêineres é derivada das propriedades das conexões, conforme definido na Tabela 3.

Os valores das propriedades Tamanho Mínimo Estimado e Tamanho Máximo Estimado de um contêiner capturam, como o nome indica, os tamanhos mínimo e máximo estimados para o contêiner. Naturalmente, estas propriedades são importantes para o processo de otimização. A Seção 6.3.1discutirá em detalhe implementações para os tipos de contêineres e estimativas para os valores das propriedades Tamanho Mínimo Estimado e Tamanho Máximo Estimado.

A propriedade Tipo de Acesso aplica-se apenas a recursos. Um recurso é *privado* quando pertence a apenas uma comunidade fechada de pesquisadores, em oposição a um recurso *público*, que está aberto a todos os pesquisadores.

Tabela 3 – Tipos de contêineres.

Tipo	Definição
gradativo	todas as conexões de entrada ou saída do contêiner são gradativas
não-gradativo	todas as conexões de entrada ou saída do contêiner são não-gradativas
misto	existe uma conexão de entrada ou saída do contêiner que é gradativa e existe uma conexão de entrada ou saída do contêiner que é não gradativa

A propriedade Formato modela o formato de representação dos dados no contêiner. Por exemplo, o Quadro 5 e o Quadro 6 apresentam uma sequência de nucleotídeos, pertencente ao banco de dados NR, no formato original de um

registro do Genbank [NCBI, 2004e] e no formato FASTA [NCBI, 2004d], respectivamente. Além destes, o NCBI ainda disponibiliza os dados do Genbank em outros formatos, como XML [NCBI, 2004g]. Normalmente, os programas exigem contêineres mantendo dados em um formato bem definido. Por exemplo, o BLASTP aceita como entrada apenas bancos de dados no formato FASTA, e gera resultados em formatos como HTML e XML [NCBI, 2004b].

```
LOCUS       ABCRRAA                 118 bp    rRNA    linear   BCT 13-DEC-1995
DEFINITION  Acetobacter sp. (strain MB 58) 5S ribosomal RNA, complete sequence.
ACCESSION   M34766
VERSION     M34766.1  GI:173603
KEYWORDS    5S ribosomal RNA.
SOURCE      Acetobacter sp.
  ORGANISM  Acetobacter sp.
            Bacteria; Proteobacteria; Alphaproteobacteria; Rhodospirillales;
            Acetobacteraceae; Acetobacter.
REFERENCE   1  (bases 1 to 118)
  AUTHORS   Bulygina,E.S., Galchenko,V.F., Govorukhina,N.I., Netrusov,A.I.,
            Nikitin,D.I., Trotsenko,Y.A. and Chumakov,K.M.
  TITLE     Taxonomic studies on methylotrophic bacteria by 5S ribosomal RNA
            sequencing
  JOURNAL   J. Gen. Microbiol. 136 (Pt 3), 441-446 (1990)
  MEDLINE   90362026
   PUBMED   2391487
COMMENT     Original source text: Acetobacter sp (strain MB 58) rRNA.
FEATURES             Location/Qualifiers
     source          1..118
                     /organism="Acetobacter sp."
                     /mol_type="rRNA"
                     /strain="MB 58"
                     /db_xref="taxon:440"
     rRNA            1..118
                     /product="5S ribosomal RNA"
ORIGIN
        1 gatctggtgg ccatggcggg agcaaatcag ccgatcccat cccgaactcg gccgtcaaat
       61 gccccagcgc ccatgatact ctgcctcaag gcacggaaaa gtcggtcgcc gccagayy
//
```

Quadro 5. Sequência de nucleotídeos no formato do Genbank.

```
>gi|173603|gb|M34766.1|ABCRRAA Acetobacter sp. (strain MB 58) 5S ribosomal RNA,
complete sequence
GATCTGGTGGCCATGGCGGGAGCAAATCAGCCGATCCCATCCCGAACTCGGCCGTCAAATGCCCCAGCGC
CCATGATACTCTGCCTCAAGGCACGGAAAAGTCGGTCGCCGCCAGAYY
```

Quadro 6. Sequência de nucleotídeos no formato FASTA.

3.5.2.2 Propriedades de Qualidade

Os contêineres também possuem propriedades de qualidade. Porém, estas propriedades são mais difíceis de caracterizar do que as propriedades de qualidade dos processos.

Por exemplo, ser *curado* ou *redundante* são qualidades dos recursos. O Swiss-Prot é um banco de dados curado, o que significa que seus dados são avaliados por um pesquisador, diferentemente do TrEMBL [Boeckmann, 2003], cujas sequências são geradas automaticamente através da tradução das sequências de nucleotídeos armazenadas no EMBL e que ainda não estão no Swiss-Prot.

Outro exemplo, o número de bases de uma sequência pode ser considerada uma qualidade das instâncias de dados de entrada ou saída. O número de bases de um contig (instância de dado de saída de uma classe de instâncias de processos de montagem de fragmentos) pode ajudar o pesquisador a definir a prioridade na análise dos seus dados. Os pesquisadores podem preferir analisar os contigs com tamanhos maiores, já que os menores têm mais chance de serem montados em contigs maiores quando novos *reads* forem sequenciados.

Cada base de um um contig também está associada a uma qualidade, que indica a chance da base ser realmente a que foi definida. Os pesquisadores podem definir trechos do contig que devem ser sequenciados novamente por não terem bases com qualidades suficientemente confiáveis.

3.6 Projetos

A classe Projeto possui instâncias, chamadas de *projetos*. Esta classe possui duas sub-classes, EST e Completo, cujas instâncias são chamadas de *projetos EST* e *projetos completos*.

Os projetos ESTs são mais comuns para genomas de eucariotos, e os projetos completos para genomas de procariotos. De fato, os projetos ESTs são mais apropriados quando o objetivo é obter somente as sequências codificantes, reduzindo tempo e custo, o que geralmente é o caso dos genomas de eucariotos, compostos na sua maioria de sequências não codificantes.

Pode-se definir outras sub-classes de Projeto, caso seja necessário distinguir projetos, por exemplo, pelos organismos que estão sendo estudados.

A determinação da classe de um projeto é importante porque influencia a escolha dos processos. Por exemplo, Phrap e CAP3 são ambos processos da classe montagem de fragmentos, mas o Phrap é a melhor escolha para projetos genoma completos, enquanto o CAP3 é mais adequado para projetos genoma de ESTs.

3.7 Comentários Finais

Este capítulo apresentou uma ontologia de processos de Bioinformática que direciona o sistema de gerência de análises em biossequências. A ontologia possui descrições das classes relevantes ao domínio de análise em biossequências, suas propriedades e como elas se relacionam. As principais classes da ontologia referem-se a processos, recursos, dados e projetos comumente envolvidos em análises de biossequências.

Outros aspectos da ontologia serão considerados nos próximos capítulos, pois são particulares a ambientes de trabalho em que o SGABio será implementado.

Como poderá ser conferido posteriormente, a ontologia é a base de conhecimento do SGABio, permitindo que o sistema possa auxiliar os pesquisadores a definir, redefinir, validar, otimizar e executar o workflow de forma adequada.