Trabalho Prático 1 - Expressões lógicas e satisfabilidade

Descrição do problema

Problemas de lógica são clássicos e possuem inúmeras variações. Neste trabalho iremos lidar com dois deles, um bem simples e o outro nem tanto. O primeiro problema será a avaliação de uma expressão lógica e o segundo será uma variação de um dos problemas mais clássicos da computação: o problema da satisfabilidade. Mas antes de mais nada vamos entender o objeto o qual iremos trabalhar.

Expressões lógicas

Iremos trabalhar com variáveis binárias e operadores bem conhecidos por nós: "E"($\wedge$), "OU"($\vee$) e "NÃO"($\neg$). Uma expressão lógica nada mais é que algumas variáveis sujeitas a esses operadores. Por exemplo:

$$(A \wedge B) \vee (\neg C \wedge D)$$

Agora vamos introduzir outros dois operadores, que chamamos **quantificadores**. Com eles nossas expressões se tornam capazes de expressar uma quantidade muito maior de objetos, eles são: "EXISTE"($\exists$) e "PARA TODO"($\forall$). Note que estes funcionam de uma forma um pouco diferente, uma vez que eles falam sobre valores que as variáveis *poderiam* assumir. Exemplo:

$$\exists A, \exists C, (A \vee B) \wedge (\neg C \vee D)$$

A expressão acima pode ser lida como: "Existem A e C tais que $(A \vee B) \wedge (\neg C \vee D)$ é verdadeira?". Note que se $A = 1$ e $C = 0$, temos que $(A \vee B) \wedge (\neg C \vee D) = 1$ independentemente dos valores de B e D.

Avaliação de expressões

O primeiro problema que iremos trabalhar é o problema da avaliação de expressões lógicas. Este problema consiste em: Você receberá uma expressão lógica $\phi(x_1, x_2, \ldots, x_n)$ onde $x_1, x_2, \ldots, x_n$ são as n variáveis que aparecem na expressão e uma valoração para elas. A resposta consiste em simplesmente substituir as variáveis por seus valores e calcular se com essa valoração ϕ vale 0 ou 1. Note que assim como em operações aritméticas, a resolução dessas operações deve seguir a seguinte ordem de precedência:

1. Precedência delimitada por "()": Assim como em operações aritméticas, as operações delimitadas por parêntesis devem ser resolvidas primeiro.
2. Operador de negação: O operador de negação tem prioridade sobre os outros.
3. Conjunção: O operador $\wedge$ é análogo a multiplicação, portanto tem precedência.
4. Disjunção: O operador $\vee$ é análogo a soma, e é o operador de menor prioridade.

Exemplo 1: Calcule $\phi(1,0,0)$ onde ϕ é descrita por:

$$\phi(A,B,C) = A \vee B \wedge C$$

Pela ordem de precedência, devemos resolver $B \wedge C$ primeiro.

$$\phi(A,B,C) = A \vee B \wedge C$$

$$\phi(1,0,0) = 1 \vee 0 \wedge 0$$

$$\phi(1,0,0) = 1 \vee 0$$

$$\phi(1,0,0) = 1$$

Note que se a relação de precedência não fosse respeitada o resultado seria diferente.

$$\phi(A,B,C) = A \vee B \wedge C$$

$$\phi(1,0,0) = 1 \vee 0 \wedge 0$$

$$\phi(1,0,0) = 1 \wedge 0$$

$$\phi(1,0,0) = 0$$

Exemplo 2: Calcule $\phi(1,0,1)$ onde ϕ é descrita por:

$$\phi(A,B,C) = \neg A \vee \neg(B \wedge C)$$

$$\phi(1,0,1) = \neg 1 \vee \neg(0 \wedge 1)$$

$$\phi(1,0,1) = 0 \vee \neg(0 \wedge 1)$$

$$\phi(1,0,1) = 0 \vee \neg(0)$$

$$\phi(1,0,1) = 0 \vee \neg 0$$

$$\phi(1,0,1) = 0 \vee 1$$

$$\phi(1,0,1) = 1$$

Observação: Os **quantificadores** não serão utilizados nas expressões a serem avaliadas no TP1.

Satisfabilidade

Este problema é um pouco diferente do anterior. Agora não estamos interessados em uma valoração específica das variáveis, mas se se **existe** uma valoração que faz com que a expressão resulte em 1. Utilizando o formalismo da seção anterior ele poderia ser descrito da forma:

$$\exists x_1, x_2, \ldots, x_n, \phi(x_1, x_2, \ldots, x_n)$$

Nós iremos estudar uma restrição desse problema. Nela apenas algumas variáveis serão quantificadas, mais precisamente no máximo **cinco** variáveis. No entanto elas poderão ser quantificadas tanto com $\exists$ quanto com $\forall$. Todas as outras variáveis terão valores pré-estabelecidos, assim como no problema anterior.

Exemplo 1: Existe A tal que $\phi(A, 0, 0)$ é verdadeira? A expressão ϕ é descrita por:

$$\phi(A, B, C) = A \vee B \wedge C$$

Note que $\phi(0, 0, 0) = 0$, no entanto $\phi(1, 0, 0) = 1$. Dessa forma a resposta é verdadeira, quando $A = 1$.

Exemplo 2: Existe B tal que $\phi(0, B, 0)$ é verdadeira? A expressão ϕ é descrita por:

$$\phi(A, B, C) = A \vee B \wedge C$$

Neste caso tanto $\phi(0, 0, 0)$ quanto $\phi(0, 1, 0)$ valem 0. Portanto a resposta é falsa.

Exemplo 3: Existe A tal que $\phi(A, 1, 1)$ é verdadeira? A expressão ϕ é descrita por:

$$\phi(A, B, C) = A \vee B \wedge C$$

Neste caso o valor de A é irrelevante, pois tanto $\phi(0, 1, 1)$ quanto $\phi(1, 1, 1)$ valem 1. Logo a resposta é verdadeira.

Vamos repetir todos os exemplos, mas agora trocando o quantificador!

Exemplo 4: Para todo A, $\phi(A, 0, 0)$ é verdadeira? A expressão ϕ é descrita por:

$$\phi(A, B, C) = A \vee B \wedge C$$

Note que $\phi(0, 0, 0) = 0$, no entanto $\phi(1, 0, 0) = 1$. Dessa forma a resposta é falsa, pois quando $A = 0$ a expressão é falsa.

Exemplo 5: Para todo B, $\phi(0, B, 0)$ é verdadeira? A expressão ϕ é descrita por:

$$\phi(A, B, C) = A \vee B \wedge C$$

Neste caso tanto $\phi(0, 0, 0)$ quanto $\phi(0, 1, 0)$ valem 0. Portanto a resposta é falsa.

Exemplo 6: Para todo A, $\phi(A, 1, 1)$ é verdadeira? A expressão ϕ é descrita por:

$$\phi(A, B, C) = A \vee B \wedge C$$

Neste caso o valor de A é irrelevante, pois tanto $\phi(0, 1, 1)$ quanto $\phi(1, 1, 1)$ valem 1. Logo a resposta é verdadeira.

Note que os dois problemas estão relacionados de alguma forma, pois estamos interessados em encontrar uma valoração específica dentre todas as possíveis. Pensando em um exemplo um pouco mais genérico se tivermos:

$$\exists x_2, \forall x_3, \phi(0, x_2, x_3, 1)$$

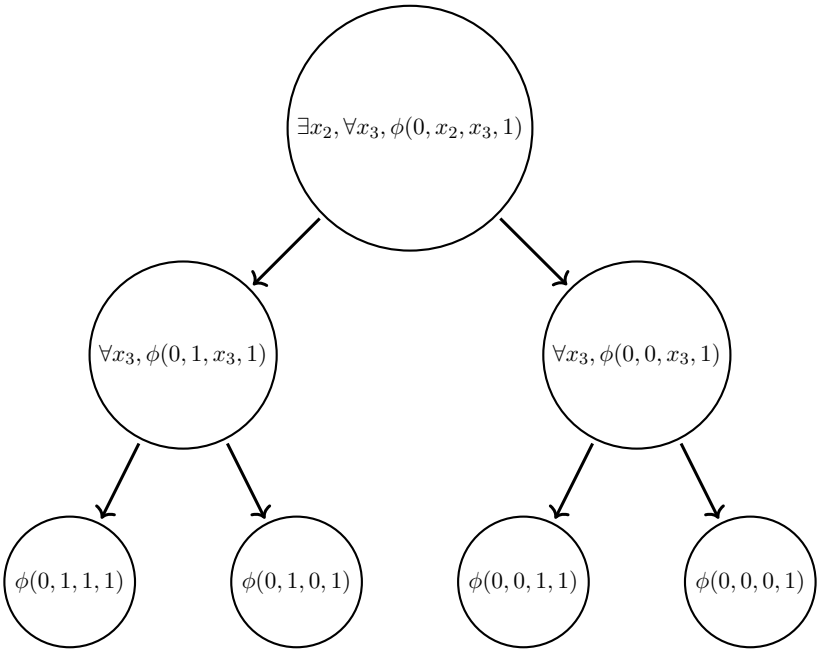

Figura 1: Possíveis soluções para $\exists x_2, \forall x_3, \phi(0, x_2, x_3, 1)$

Observe a figura 1 e os exemplos apresentados. Como as soluções parciais se relacionam? Entender essa relação é crucial para a resolução deste problema.

Observação: É imprescindível que a resolução dos quantificadores seja feita na ordem em que eles aparecem! Teste alguns exemplos para se convencer disso caso necessário.

O que você deve implementar

Avaliação de expressões

Para este problema você deve implementar um avaliador de expressões lógicas. Sua entrada serão duas strings p e s. A string p contém a representação de uma fórmula lógica e seu tamanho máximo é da ordem de 10^6 caracteres. A string s é uma string binária que contém uma valoração para as variáveis presentes na fórmula. O tamanho de s é no máximo 100.

Na representação da fórmula as variáveis são indicadas por inteiros, de forma que a variável x_i é representada pelo inteiro i. Os operadores $\wedge$, $\vee$ e $\neg$ serão representados, respectivamente por "&", "|"e "$\sim$". Todos os símbolos da fórmula, incluindo parêntesis estarão separados por um espaço em branco.

A string s estará codificada de forma que o valor que deverá ser associado a variável x_i seja o i-ésimo caractere da string.

A saída deve ser apenas um inteiro, "0"ou "1", contendo o resultado da avaliação da expressão. A saída deve ser uma impressão na tela, e deve ser encerrada com uma quebra de linha.

Exemplo: A fórmula é: $\phi(x_0, x_1, x_2) = x_1 \wedge \neg x_2 \vee (x_1 \wedge x_0)$ e desejamos calcular $\phi(0, 0, 1)$. As strings p e s serão, respectivamente:

```
p := 1 & ~ 2 | ( 1 & 0 )

s := 001
```

OBSERVAÇÃO IMPORTANTE: Para a resolução deste problema é **vetado** o uso de recursão. Você deverá utilizar alguma estrutura de dados vista em sala para resolver as precedências!

Satisfabilidade

Neste problema você deverá encontrar, caso exista, uma valoração que satisfaça a expressão. A entrada será semelhante a do problema anterior, a principal diferença está na string s que não será mais uma string binária. Os quantificadores estarão codificados nela. caso o i-ésimo caractere da string s seja o caractere "e", o quantificador da variável x_i é um $\exists$. De forma análoga se o caractere for um "a", o quantificador será um $\forall$. Os quantificadores devem ser avaliados na ordem em que aparecem nessa string.
A saída consistirá de um inteiro "0"caso não exista valoração válida ou "1"caso exista. Se a resposta for verdadeira você deve informar uma string com a valoração válida (se o valor de uma variável for irrelevante você deve imprimir o caractere "a"). A saída deve ser uma impressão na tela, o caractere de resposta deve estar separado por um espaço em branco da string contendo a valoração caso ela exista, e deve ser encerrada com uma quebra de linha.

OBSERVAÇÃO IMPORTANTE: O uso de recursão nessa parte é permitido caso seja necessário para realizar operações com sua estrutura de dados. Não é permitido o uso de recursão para simular as estruturas de dados. Todas as instâncias possuirão solução **única** na formatação solicitada.

Entrada

Seu programa deve receber três parâmetros de linha de comando. O primeiro será:

- -a: se o problema a ser resolvido é o de avaliação de expressões.
- -s: Se o problema a ser resolvido é o de satisfabilidade.

O segundo sempre será uma string contendo a representação da fórmula e o terceiro outra string contendo a valoração.

Exemplos

Exemplo 1:

```
./tp1.out -a "0 | 1" 01
```

Saída esperada:

```
1
```

Exemplo 2:

```
./tp1.out -a "0 | 1 & 2" 010
```

Saída esperada:

```
0
```

Exemplo 3:

```
./tp1.out -a "~ ( 0 | 1 ) & 2" 101
```

Saída esperada:

```
0
```

Exemplo 4:

```
./tp1.out -s "0 | 1 & 2" 0e0
```

Saída esperada:

```
0
```

Exemplo 5:

```
./tp1.out -s "0 | 1 & 2" e00
```

Saída esperada:

```
1 100
```

Exemplo 6:

```
./tp1.out -s "0 | 1 & 2" e11
```

Saída esperada:

```
1 a11
```

Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional, assim como as análises dos resultados.
7. **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
9. Número máximo de páginas incluindo a capa: 10

No item 4, não se esqueça de justificar qual o tamanho da entrada. Compare a ordem de complexidade dos dois problemas e discorra sobre o resultado. O que aconteceria com seu algoritmo da satisfabilidade caso a restrição de apenas 5 variáveis quantificadas fosse removida? Como se comportaria agora sua complexidade de tempo e espaço?

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

Como será feita a entrega

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim a pasta **obj** deve estar vazia. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp1.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no moodle. Você também terá a disposição uma VPL de testes para conferir se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não vale pontos e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no moodle.
A documentação será entregue em uma atividade separada designada para tal no Moodle. A entrega deve ser feita em um único arquivo com extensão .pdf, com nomenclatura nome_sobrenome_matricula.pdf, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

Avaliação

O trabalho será avaliado de acordo com:

- Definição e implementação das estruturas de dados e funções - (30% da nota total)
- Corretude na execução dos casos de teste - (20% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Cumprimento total da especificação - (5% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

[1]Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

FAQ (Frequently asked Questions)

1. Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO
2. Posso utilizar smart pointers? NÃO.
3. Posso utilizar o tipo String? SIM.
4. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO
5. Posso utilizar alguma biblioteca para tratar exceções? SIM.
6. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
7. As análises e apresentação dos resultados são importantes na documentação? SIM.
8. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
9. Posso fazer o trabalho em dupla ou em grupo? NÃO
10. Posso trocar informações com os colegas sobre a teoria? SIM.
11. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
12. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.