Projeto Tiny-OS

Claudio Emanoel Barbosa Lima
Universidade Federal de Santa Catarina
Centro Tecnológico de Joinville
Joinville, Brasil
claudiiolima@gmail.com

***Resumo*—Este documento apresenta o modelo de um sistema operacional e explora as propriedades de *deadlock freedom* (*safety*) e *liveness*. A implementação, funcionamento, características e resultados discutidos ao longo do texto foram desenvolvidos e obtidos utilizando a ferramenta de modelagem, validação e verificação de sistemas TAPAAL [1].**

***Palavras-Chave*—Testabilidade, Modelagem, TAPAAL**

I. Introdução

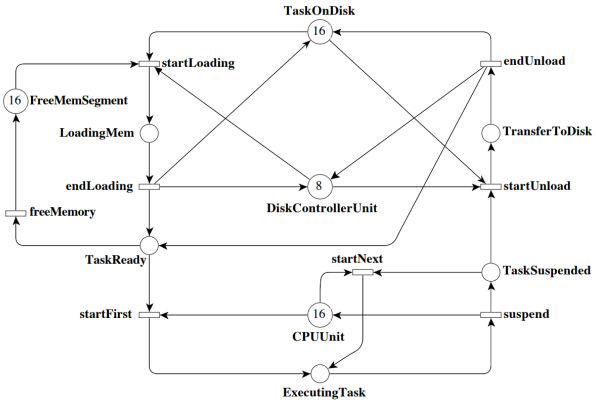
O sistema operacional é composto por unidades de disco, processamento, e memória, além das tarefas que serão executadas nesse sistema.

Os parâmetros, M (Segmentos de memória), T (tarefas) D (Controladores de Disco) e C (Núcleos), do modelo respeitam as seguintes equações:

$$\begin{aligned} M = T = MT \\ D = DC \\ C = 2DC \end{aligned} \tag{1}$$

A Figura 1, ilustra o modelo na configuração $MT = 16$ e $DC = 8$, utilizando redes de Petri.

Figura 1. SmallOperatingSystem



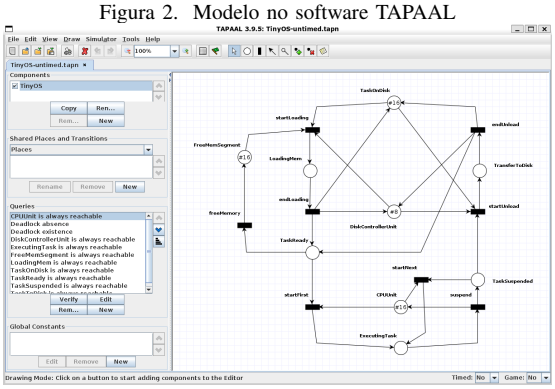
Fonte: Kondor, F. (2015) [2].

II. Desenvolvimento

O *software* de modelagem utilizado, TAPAAL, descreve, simula, e verifica redes de Petri, portanto não houve desenvolvimento de modelagem, uma vez que o sistema proposto foi representado usando uma rede de Petri.

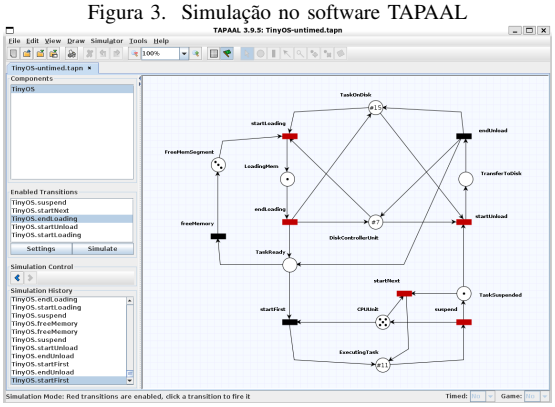
Após transcrever o modelo para o *software*, Figura 2, é possível simular o comportamento dinâmico de maneira aleatória ou escolhendo as transições sensíveis, a Figura 3 mostra o sistema após 1052 transições, é possível visualizar através das cores a sensibilidade de cada transição, sendo as de coloração vermelhas suscetíveis a estímulos.

Figura 2. Modelo no software TAPAAL



Fonte: Autor (2023).

Figura 3. Simulação no software TAPAAL



Fonte: Autor (2023).

Para realizar a verificação do modelo utilizou-se conceitos de lógica de arvore computacional (CTL - *Computation Tree Logic*), fazendo o uso de quantificadores e operadores [3].

Quantificadores de caminho:

- A - Para todos caminhos (inevitavelmente).
- E - Existe ao menos um caminho (possivelmente).

Operadores de tempo:

- $G\ p$ - p é válido em todos os estados futuros (globalmente).
- $F\ p$ - p é válido em algum estado futuro.
- $X\ p$ - p é válido no próximo estado.
- $p\ U\ q$ - p é válido até que q ser válido.

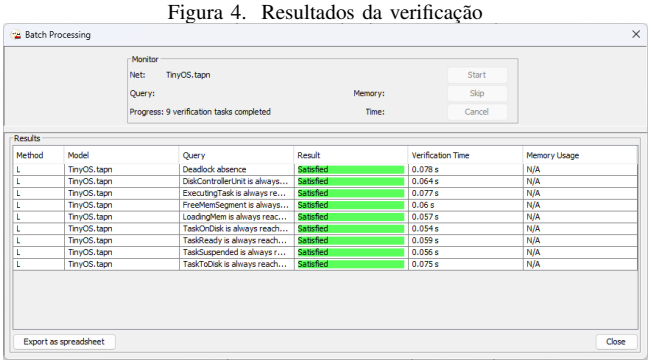
As seguintes fórmulas foram compostas para verificar propriedades de segurança (*safety*) e vivacidade (*liveness*):

- $AG\ !(deadlock)$
- $AF\ (EF\ CPUUnit\ \neq\ 0)$
- $AF\ (EF\ DiskControllerUnit\ \neq\ 0)$
- $AF\ (EF\ ExecutingTask\ \neq\ 0)$
- $AF\ (EF\ FreeMemSegment\ \neq\ 0)$
- $AF\ (EF\ LoadingMem\ \neq\ 0)$
- $AF\ (EF\ TaskOnDisk\ \neq\ 0)$
- $AF\ (EF\ TaskReady\ \neq\ 0)$
- $AF\ (EF\ TaskSuspended\ \neq\ 0)$
- $AF\ (EF\ TransferToDisk\ \neq\ 0)$

III. Resultados

Obteve-se sucesso ao executar a ferramenta de verificação do software TAPAAL para as propriedades propostas. O resultado e seus detalhes podem ser visualizados na Figura 4.

Figura 4. Resultados da verificação



Fonte: Autor (2023).

IV. Conclusão

O *software* utilizado apresentou comportamento estável e esperado, seja para o modo de simulação, onde foi possível visualizar o comportamento do sistema de acordo com o tempo de execução, assim como o verificador e seus resultados.

Referências

[1] Department of Computer Science at Aalborg University in Denmark., "TAPAAL: Tool for verification of timed-arc petri nets," 2023. [Online]. Available: https://www.tapaal.net/

[2] "Model checking contest @ petri nets 2015," https://mcc.lip6.fr/2015/models.php, 2015.

[3] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.