A) CAPÍTULO — Validação Experimental do Core

Você pode copiar este texto direto para o relatório/TCC.

6. Validação Experimental do Core

A validação experimental do núcleo computacional (core) do projeto HyperVLM teve como objetivo verificar propriedades fundamentais de correção, robustez e previsibilidade da execução, sem dependência de interfaces gráficas, motores externos ou aplicações finais.

Todos os experimentos foram conduzidos utilizando exclusivamente o core isolado, compilado como biblioteca estática e executado por meio de uma ferramenta de linha de comando dedicada.

6.1 Metodologia de Validação

A validação foi estruturada em três classes de testes:

Isolamento estrutural

Execução funcional mínima

Determinismo e tratamento de falhas

O ambiente de testes consistiu em sistema Linux, compilador GNU C++ e sistema de build CMake, com execução direta via terminal.

6.2 Isolamento do Core

O core foi compilado de forma isolada, sem inclusão de módulos de renderização, path tracing ou motores gráficos. Para isso, foi criado um alvo específico de build contendo apenas os módulos essenciais:

Análise léxica (Lexer)

Análise sintática (Parser)

Construção de representação intermediária (IR)

Máquina virtual (VM)

Ambiente de execução (DRE)

A compilação isolada foi concluída com sucesso, demonstrando que o core não depende de componentes externos para sua construção ou execução.

Resultado: isolamento estrutural comprovado.

6.3 Execução Funcional

Um programa mínimo válido foi utilizado como entrada para o core. O processo envolveu duas fases distintas:

Compilação da entrada para um artefato intermediário

Execução controlada pela máquina virtual

A execução apresentou comportamento incremental, com avanço explícito por etapas da VM, sem falhas ou comportamentos indefinidos.

Resultado: execução funcional mínima validada.

6.4 Determinismo

O mesmo programa de entrada foi executado múltiplas vezes sob as mesmas condições. As saídas produzidas foram comparadas byte a byte.

Não foram observadas diferenças entre as execuções sucessivas.

Resultado: o core apresenta comportamento determinístico sob entradas equivalentes.

6.5 Tratamento de Erros

Entradas sintaticamente inválidas foram fornecidas ao sistema com o objetivo de testar robustez. O core detectou o erro durante a fase de análise sintática e abortou a execução de forma segura, apresentando mensagem clara e controlada.

Não ocorreram falhas críticas, travamentos ou corrupção de estado.

Resultado: falhas são tratadas de forma previsível e segura.

6.6 Invariantes Validadas

A partir dos experimentos, foram confirmadas as seguintes invariantes do core:

Determinismo da execução

Separação entre compilação e execução

Rejeição segura de entradas inválidas

Execução incremental e controlada

Independência de interfaces e motores externos

Essas propriedades qualificam o core como um núcleo computacional confiável, apto a servir como base para extensões futuras.

7. Modelo de Estados da Máquina Virtual

A Máquina Virtual (VM) do projeto HyperVLM é responsável pela execução determinística de programas compilados para a representação intermediária do sistema. Seu funcionamento é descrito por um modelo explícito de estados, no qual cada transição é controlada e observável.

Esse modelo garante previsibilidade, isolamento de falhas e separação clara entre as fases de compilação e execução.

7.1 Visão Geral

A VM opera como uma máquina de estados finitos, na qual apenas transições válidas são permitidas. Cada estado representa uma fase bem definida do ciclo de vida de execução de um programa.

O comportamento observado experimentalmente — incluindo carregamento do artefato .vlm e execução passo a passo — corresponde diretamente a esse modelo formal.

7.2 Estados da VM

A seguir estão os estados formais definidos para a Máquina Virtual:

INIT Estado inicial da VM. Nenhum programa foi carregado.

LOAD Estado em que o artefato intermediário (.vlm) é carregado e validado.

READY Programa carregado com sucesso. A VM está pronta para iniciar a execução.

STEP Execução incremental de uma instrução ou bloco lógico. Esse estado pode ocorrer repetidamente.

HALT Estado final de execução normal. O programa terminou sem erros.

ERROR Estado terminal atingido quando ocorre erro sintático, semântico ou violação de invariante.

7.3 Diagrama de Estados +——+ | INIT | +——+ | v +——+ | LOAD | +——+ | +—+—+ | | v v +———+ +———+ | READY | | ERROR | +———+ +———+ | v +———+ | STEP |<——+ +———+ | | | v | +———+ | | HALT |——-+ +———+

7.4 Transições de Estado Estado Atual Evento Próximo Estado INIT Inicialização da VM LOAD LOAD Artefato válido carregado READY LOAD Erro de leitura ou validação ERROR READY Início da execução STEP STEP Instrução válida executada STEP STEP Fim do programa HALT STEP Erro de execução ERROR
7.5 Propriedades do Modelo

O modelo de estados da VM garante as seguintes propriedades:

Determinismo: a execução segue sempre a mesma sequência de estados para entradas equivalentes.

Execução incremental: a VM avança por passos discretos, facilitando inspeção e controle.

Isolamento de falhas: qualquer erro leva a um estado terminal seguro.

Separação de fases: compilação e execução não se sobrepõem no modelo de estados.

7.6 Relação com a Validação Experimental

Os testes realizados confirmaram empiricamente esse modelo. Em particular:

A mensagem [VM] loading corresponde à transição INIT → LOAD.

As mensagens [VM] step correspondem às iterações no estado STEP.

A ausência de falhas confirma transições válidas até HALT.

Erros de parsing conduzem diretamente ao estado ERROR.

Assim, o modelo teórico apresentado está em conformidade com o comportamento observado do sistema.

7.7 Considerações Finais

O modelo de estados da Máquina Virtual fornece uma base formal para o entendimento da execução do core do HyperVLM, sendo essencial para análise de corretude, extensibilidade futura e justificativa de propriedades como determinismo e robustez.

8. Arquitetura do Core

O core do projeto HyperVLM foi projetado como um núcleo computacional independente, responsável por análise, transformação e execução de programas, sem dependência de interfaces gráficas ou motores externos.

8.1 Visão Geral da Arquitetura

A arquitetura segue uma separação em camadas bem definidas:

- Entrada textual (programa fonte)
- Análise léxica
- Análise sintática
- Representação intermediária (IR)
- Máquina Virtual (VM)
- Ambiente de execução (DRE)

Cada camada comunica-se apenas com a camada adjacente, evitando acoplamento indevido.

8.2 Módulos Principais

- **Lexer**: converte texto em tokens
- **Parser**: constrói a árvore sintática
- **IRBuilder**: gera a representação intermediária
- **VM**: executa o programa de forma determinística
- **DRE**: gerencia o ambiente de execução

8.3 Isolamento Arquitetural

O core pode ser compilado e executado isoladamente, como demonstrado nos testes experimentais, garantindo sua reutilização e extensibilidade.

8.4 Extensibilidade

Novos módulos podem ser integrados ao core sem alteração das invariantes fundamentais, preservando corretude e determinismo.

]## 9. Metodologia de Desenvolvimento

O desenvolvimento do HyperVLM seguiu uma metodologia incremental e experimental, baseada em validação contínua.

9.1 Abordagem Incremental

Cada componente do core foi implementado e testado isoladamente antes de integração completa.

9.2 Validação Contínua

Testes manuais e automatizados foram utilizados para verificar determinismo, isolamento e robustez.

9.3 Uso de Evidência Empírica

Decisões arquiteturais foram fundamentadas em resultados observáveis, evitando suposições não verificadas.

9.4 Controle de Complexidade

A complexidade foi controlada por meio de separação de responsabilidades e isolamento de módulos.

10. Checklist de Avaliação e Defesa

10.1 Perguntas Esperadas

- O sistema é determinístico?
  Sim, comprovado experimentalmente.
- O core é independente?
  Sim, compilável e executável isoladamente.
- Existe modelo formal?
  Sim, modelo de estados da VM.

10.2 Diferenciais Técnicos

- Core isolado
- VM própria
- Execução incremental
- Validação empírica documentada

10.3 Limitações Conhecidas

- Ausência de interface gráfica
- Foco em núcleo computacional

Essas limitações são decisões de projeto, não falhas.

11. Conclusão

Este trabalho apresentou o desenvolvimento e validação de um núcleo computacional próprio, com foco em determinismo, isolamento e execução controlada.

Os resultados experimentais confirmam que o core do HyperVLM atende aos requisitos propostos, constituindo uma base sólida para extensões futuras.

O projeto cumpre seus objetivos acadêmicos e técnicos, demonstrando viabilidade, corretude e maturidade arquitetural.