# Code Documentation: src

UMEDCTA Repository

December 3, 2025

## Contents

# 1 src/DialecticalInterpreter.jsx

```jsx
import React, { useState } from 'react';
import { AlertCircle, Sparkles, GitBranch, CheckCircle, XCircle, Loader2, Copy } from 'lucide-react';

// Helper function to call our backend API
const callAnthropic = async (messages, maxTokens = 2000) => {
  const response = await fetch('http://localhost:3001/api/anthropic', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      model: 'claude-sonnet-4-20250514',
      max_tokens: maxTokens,
      messages
    })
  });

  if (!response.ok) {
    const error = await response.json();
    throw new Error(error.error || 'API request failed');
  }

  return response.json();
};

const DialecticalInterpreter = () => {
  const [inputText, setInputText] = useState('');
  const [interpretation, setInterpretation] = useState(null);
  const [isProcessing, setIsProcessing] = useState(false);
  const [evolutionHistory, setEvolutionHistory] = useState([]);
  const [showEvolutionPanel, setShowEvolutionPanel] = useState(false);

  // Conversation state for follow-ups
  const [conversationHistory, setConversationHistory] = useState([]);
  const [followUpQuestion, setFollowUpQuestion] = useState('');
  const [showFollowUp, setShowFollowUp] = useState(false);

  // Current axiom set (starts with base PML axioms)
  const [axiomSet, setAxiomSet] = useState([
    {
      id: 'base_rhythm',
      content: 's(u) => s(comp_nec a)',
      source: 'core',
      type: 'material',
      context: 'Fundamental dialectical rhythm: unity necessarily generates tension'
    },
    {
      id: 'sublation',
      content: 's(lg) => s(exp_nec u_prime)',
      source: 'core',
      type: 'material',
      context: 'Letting go necessarily produces new unity'
    },
    {
      id: 'oobleck',
      content: 's(comp_nec P) => o(comp_nec P)',
      source: 'core',
      type: 'material',
      context: 'Subjective compression crystallizes objective content'
    }
  ]);

  // Iteration tracking: how many times has this text been read?
  const [iterationDepth, setIterationDepth] = useState(0);
```

```
  const [formalizedConcepts, setFormalizedConcepts] = useState([]);

  const processText = async () => {
    setIsProcessing(true);
    setInterpretation(null);

    // Check if this is a re-read of the same text
    const previousRead = conversationHistory.find(
      item => item.type === 'interpretation' && item.content.original === inputText
    );
    const isRereading = !!previousRead;
    const currentIteration = isRereading ? iterationDepth + 1 : 1;

    try {
      // Phase 1: Parse text into PML
      const iterationContext = isRereading ? `
ITERATION DEPTH: ${currentIteration} (This is a RE-READING)

CRITICAL: On second+ readings, established interpretations become PART OF the phenomenology.
- First reading: All material inferences (discovering what concepts mean)
- Second+ reading: Some concepts are now FORMALIZED (structural scaffolding, not discovery)
- The reader brings prior understanding as background assumptions

Previous interpretation of this text:
${JSON.stringify(previousRead?.content.logic.interpretation)}

Key concepts from previous read:
${JSON.stringify(previousRead?.content.pml.key_concepts)}

These concepts now operate as FORMAL structure rather than material discovery.` : `
ITERATION DEPTH: 1 (FIRST READING)

This is a first encounter with the text. The reader:
- Discovers concepts through material inference (content-based reasoning)
- Experiences genuine novelty and surprise
- Builds understanding from scratch without prior scaffolding`;

      const parsePrompt = `You are a philosophical interpreter using Polarized Modal Logic (PML).

CRITICAL FRAMING: PML tracks the PHENOMENOLOGY OF READING - the temporal, embodied experience of working
↪  through philosophical text. It models:
- How tension builds as you encounter concepts (compression ↓)
- How understanding releases when connections form (expansion ↑)
- The cognitive resources consumed in the reading process
- The subjective experience of the dialectical rhythm

${iterationContext}

PML Vocabulary:
- Three modes: s(P) = subjective experience, o(P) = objective claim, n(P) = normative commitment
- Four modalities: comp_nec(P) = necessary compression (↓), exp_nec(P) = necessary expansion (↑),
  comp_poss(P) = possible compression, exp_poss(P) = possible expansion
- Dialectical rhythm: u → comp_nec(a) → exp_poss(lg) → exp_nec(u_prime)
  (unity → tension → possibility of release → new understanding)

Current Axiom Set (including evolved axioms):
${axiomSet.map(a => `- [${a.type}] ${a.content} // ${a.context}`).join('\n')}

Formalized Concepts (operate as structural scaffolding on re-reads):
${formalizedConcepts.length > 0 ? formalizedConcepts.join(', ') : 'None yet'}

Text to interpret AS A TEMPORAL UNFOLDING:
"${inputText}"

Task: Formalize how a reader EXPERIENCES this text moving through it sequentially. Track:
```

```
1. Initial subjective state (what's your starting point?)
2. Compressive moments (where does tension/confusion arise?)
3. Expansive moments (where does understanding open up?)
4. The temporal sequence of dialectical transitions
5. Resource costs (where is the text cognitively demanding?)
6. ${isRereading ? 'FORMALIZATION: What concepts from prior reads now operate as formal structure?' : 'DISCOVERY: What concepts are being discovered for the first time?'}

Respond ONLY with valid JSON (no markdown):
{
  "formalizations": [
    {"step": "initial_state", "pml": "...", "explanation": "reader's starting point", "temporal_moment": "beginning", "inference_type": "material|formal"},
    {"step": "compression", "pml": "...", "explanation": "where tension builds", "temporal_moment": "middle", "inference_type": "material|formal"},
    {"step": "expansion", "pml": "...", "explanation": "where understanding releases", "temporal_moment": "late", "inference_type": "material|formal"}
  ],
  "reading_experience": "Overall phenomenological description of working through this text",
  "key_concepts": ["concept1", "concept2"],
  "formalized_this_iteration": ["concepts that should become formal scaffolding on next read"],
  "iteration_depth": ${currentIteration}
}`;

      const parseData = await callAnthropic([
        { role: "user", content: parsePrompt }
      ], 2000);
      let parseText = parseData.content[0].text.trim();
      parseText = parseText.replace(/```json\n?/g, "").replace(/```\n?/g, "").trim();
      const parsedPML = JSON.parse(parseText);

      // Phase 2: Run logic and generate interpretation
      const interpretPrompt = `Using the PML formalizations, generate an interpretation by tracing through the logic.

Formalizations:
${JSON.stringify(parsedPML.formalizations, null, 2)}

Available Axioms:
${axiomSet.map(a => `- ${a.content}`).join('\n')}

Apply the axioms to derive conclusions. Show each inference step.

Then, provide your philosophical interpretation of the text based on this logical structure.

Respond ONLY with valid JSON:
{
  "proof_steps": [
    {"premises": ["..."], "axiom_used": "...", "conclusion": "...", "explanation": "..."}
  ],
  "interpretation": "Your philosophical reading of the text...",
  "key_insights": ["insight1", "insight2"]
}`;

      const interpretData = await callAnthropic([
        { role: "user", content: interpretPrompt }
      ], 3000);
      let interpretText = interpretData.content[0].text.trim();
      interpretText = interpretText.replace(/```json\n?/g, "").replace(/```\n?/g, "").trim();
      const logicResult = JSON.parse(interpretText);

      // Phase 3: Critique - Compare against established readings
      const critiquePrompt = `You are a meta-critic analyzing different LEVELS OF ANALYSIS.

Original Text: "${inputText}"
```

```
187
188 Our PML Interpretation (PHENOMENOLOGICAL LEVEL - tracking the embodied reading experience):
189 ${logicResult.interpretation}
190
191 CRITICAL CONTEXT: PML tracks the TEMPORAL PHENOMENOLOGY of reading, not atemporal propositional content.
192 A "contradiction" with traditional interpretations often reveals that we're analyzing different levels:
193 - Traditional: "What does the text SAY?" (propositional, atemporal)
194 - PML: "How does it FEEL to work through this text?" (phenomenological, temporal, embodied)
195
196 Task:
197 1. What are the major scholarly interpretations of this passage? (Focus on PROPOSITIONAL content)
198 2. How does our PHENOMENOLOGICAL reading compare?
199 3. Are apparent contradictions actually tracking different levels?
200 4. What might deepen our phenomenological analysis?
201
202 DO NOT OUTPUT ANYTHING OTHER THAN VALID JSON:
203 {
204   "established_readings": [
205     {"scholar": "...", "interpretation": "...", "level": "propositional|phenomenological"}
206   ],
207   "alignment": {
208     "level_distinctions": [
209       {"apparent_contradiction": "...", "actually": "different levels - both valid", "explanation":
      ↪  "..."}
210     ],
211     "genuine_contradictions": [
212       {"issue": "...", "our_claim": "...", "standard_claim": "...", "severity": "high|medium|low"}
213     ]
214   },
215   "diagnostic": {
216     "missing_phenomenological_moves": ["what reading experiences are we not tracking?"],
217     "needed_axioms": [
218       {"proposed": "...", "rationale": "...", "addresses": "..."}
219     ],
220     "pathology_detected": "fixation|bad_infinite|none",
221     "meta_insight": "What does this text teach us about the phenomenology of reading Hegel?"
222   }
223 }`;
224
225       const critiqueData = await callAnthropic([
226         { role: "user", content: critiquePrompt }
227       ], 3000);
228       let critiqueText = critiqueData.content[0].text.trim();
229       critiqueText = critiqueText.replace(/```json\n?/g, "").replace(/```\n?/g, "").trim();
230       const critique = JSON.parse(critiqueText);
231
232       // Combine results
233       const newInterpretation = {
234         original: inputText,
235         pml: parsedPML,
236         logic: logicResult,
237         critique: critique,
238         timestamp: new Date().toISOString(),
239         iterationDepth: currentIteration
240       };
241
242       setInterpretation(newInterpretation);
243
244       // Update iteration tracking
245       if (isRereading) {
246         setIterationDepth(currentIteration);
247       } else {
248         setIterationDepth(1);
249       }
250
```

```
      // Update formalized concepts if this is a second+ reading
      if (parsedPML.formalized_this_iteration && parsedPML.formalized_this_iteration.length > 0) {
        setFormalizedConcepts([
          ...new Set([...formalizedConcepts, ...parsedPML.formalized_this_iteration])
        ]);
      }

      // Add to conversation history
      setConversationHistory([
        ...conversationHistory,
        {
          type: 'interpretation',
          content: newInterpretation
        }
      ]);

      setShowFollowUp(true);

    } catch (error) {
      console.error("Error processing text:", error);
      setInterpretation({
        error: true,
        message: error.message
      });
    }

    setIsProcessing(false);
  };

  const handleFollowUp = async () => {
    if (!followUpQuestion.trim() || !interpretation) return;

    setIsProcessing(true);

    try {
      // Build conversation context
      const contextMessages = conversationHistory.map(item => {
        if (item.type === 'interpretation') {
          return {
            role: "assistant",
            content: `I analyzed the text phenomenologically using PML and found:
            ↪  ${item.content.logic.interpretation}`
          };
        } else if (item.type === 'followup') {
          return [
            { role: "user", content: item.question },
            { role: "assistant", content: item.response }
          ];
        }
      }).flat();

      const followUpPrompt = `CONTEXT: You previously analyzed this text using PML (Polarized Modal
      ↪  Logic), which tracks the PHENOMENOLOGY OF READING - the temporal, embodied experience of
      ↪  working through text.

Original text: "${interpretation.original}"

Your PML analysis: ${interpretation.logic.interpretation}

Critique insights: ${JSON.stringify(interpretation.critique.diagnostic, null, 2)}

User's follow-up question/clarification:
"${followUpQuestion}"

Respond to their question while:
```

```
1. Maintaining focus on the PHENOMENOLOGICAL level (how the text feels to read)
2. Distinguishing between propositional content vs. reading experience when relevant
3. Suggesting refinements to the PML formalization if needed
4. Acknowledging where the logic might need evolution

Be conversational and dialectical. Help refine the interpretation through dialogue.`;

      const followUpData = await callAnthropic([
        ...contextMessages,
        { role: "user", content: followUpPrompt }
      ], 2000);
      const response = followUpData.content[0].text;

      // Add to conversation history
      setConversationHistory([
        ...conversationHistory,
        {
          type: 'followup',
          question: followUpQuestion,
          response: response,
          timestamp: new Date().toISOString()
        }
      ]);

      // Update interpretation with follow-up
      setInterpretation({
        ...interpretation,
        followUps: [...(interpretation.followUps || []), {
          question: followUpQuestion,
          response: response
        }]
      });

      setFollowUpQuestion('');

    } catch (error) {
      console.error("Error in follow-up:", error);
      alert('Error processing follow-up: ' + error.message);
    }

    setIsProcessing(false);
  };

  const accommodateContradiction = async (contradiction, proposedAxiom) => {
    setIsProcessing(true);

    try {
      // Sublation: Synthesize new axiom
      const sublationPrompt = `You detected a contradiction in our PML interpretation.

Contradiction: ${contradiction.issue}
Our claim: ${contradiction.our_claim}
Standard claim: ${contradiction.standard_claim}

Proposed axiom: ${proposedAxiom.proposed}
Rationale: ${proposedAxiom.rationale}

Refine this axiom into proper PML syntax. Ensure it:
1. Resolves the contradiction
2. Preserves existing valid inferences
3. Opens new interpretive possibilities

Respond ONLY with valid JSON:
{
  "refined_axiom": "PML syntax here",
```

```
  "integration_strategy": "How this fits with existing axioms",
  "test_implications": ["What this now lets us infer..."],
  "context": "One-sentence summary of why this axiom was needed"
}`;

      const sublationData = await callAnthropic([
        { role: "user", content: sublationPrompt }
      ], 2000);
      let sublationText = sublationData.content[0].text.trim();
      sublationText = sublationText.replace(/```json\n?/g, "").replace(/```\n?/g, "").trim();
      const refinedAxiom = JSON.parse(sublationText);

      // Add to axiom set
      const newAxiom = {
        id: `evolved_${Date.now()}`,
        content: refinedAxiom.refined_axiom,
        source: 'evolved',
        type: 'material', // New axioms start as material, may become formal through iteration
        rationale: proposedAxiom.rationale,
        addresses: contradiction.issue,
        context: refinedAxiom.context,
        timestamp: new Date().toISOString()
      };

      setAxiomSet([...axiomSet, newAxiom]);

      // Record evolution
      setEvolutionHistory([
        ...evolutionHistory,
        {
          timestamp: new Date().toISOString(),
          trigger: contradiction.issue,
          oldState: axiomSet.length + ' axioms',
          newAxiom: refinedAxiom.refined_axiom,
          synthesis: refinedAxiom.integration_strategy,
          context: refinedAxiom.context
        }
      ]);

      alert('Axiom integrated! Try reprocessing the text to see how the interpretation changes.');

    } catch (error) {
      console.error("Error in accommodation:", error);
      alert('Error during sublation: ' + error.message);
    }

    setIsProcessing(false);
  };

  // Export functions
  const copyInterpretation = () => {
    if (!interpretation) return;

    const exportText = `
=== PML PHENOMENOLOGICAL READING ===
Text: "${interpretation.original}"
Iteration: ${interpretation.iterationDepth}
Timestamp: ${new Date(interpretation.timestamp).toLocaleString()}

READING EXPERIENCE:
${interpretation.pml.reading_experience}

PML FORMALIZATIONS:
${interpretation.pml.formalizations.map(f => `
${f.step.toUpperCase()} [${f.temporal_moment}] [${f.inference_type || 'material'}]
```

```
  PML: ${f.pml}
  ${f.explanation}
`).join('\n')}

INTERPRETATION:
${interpretation.logic.interpretation}

KEY INSIGHTS:
${interpretation.logic.key_insights?.map(i => `- ${i}`).join('\n') || 'None'}

META-CRITIQUE:
${interpretation.critique.diagnostic.meta_insight || 'None'}
`;

    navigator.clipboard.writeText(exportText);
    alert('Interpretation copied to clipboard!');
  };

  const exportAxiomsAsProlog = () => {
    const prologCode = `
%% ============================================================
%% PML Axioms - Exported from Dialectical Interpreter
%% Generated: ${new Date().toLocaleString()}
%% Total Axioms: ${axiomSet.length}
%% Formalized Concepts: ${formalizedConcepts.join(', ') || 'None'}
%% ============================================================

:- module(evolved_axioms, []).
:- use_module(pml_operators).
:- multifile incompatibility_semantics:material_inference/3.

${axiomSet.map(axiom => `
%% ${axiom.context}
%% Source: ${axiom.source}, Type: ${axiom.type}
${axiom.source === 'evolved' ? `%% Added: ${new Date(axiom.timestamp).toLocaleString()}` : ''}
${axiom.rationale ? `%% Rationale: ${axiom.rationale}` : ''}
${axiom.addresses ? `%% Addresses: ${axiom.addresses}` : ''}
${convertToPrologAxiom(axiom.content)}`).join('\n\n')}
`;

    navigator.clipboard.writeText(prologCode);
    alert('Prolog axioms copied to clipboard! Save as evolved_axioms.pl and load after
    ↪  semantic_axioms.');
  };

  const convertToPrologAxiom = (axiomContent) => {
    // Parse simple axiom syntax and convert to Prolog material_inference/3
    const match = axiomContent.match(/^(.*?)\s*=>\s*(.*)$/);
    if (!match) return `%% Could not parse: ${axiomContent}`;

    const antecedent = match[1].trim();
    const consequent = match[2].trim();

    return `incompatibility_semantics:material_inference([${antecedent}], ${consequent}, true).`;
  };

  const autoAddAxiom = async (proposedAxiom) => {
    // Automatically add an axiom without going through accommodation flow
    const newAxiom = {
      id: `auto_${Date.now()}`,
      content: proposedAxiom.proposed,
      source: 'user_suggested',
      type: 'material',
      rationale: proposedAxiom.rationale,
      context: proposedAxiom.rationale.substring(0, 100), // First 100 chars
```

```
      timestamp: new Date().toISOString()
    };

    setAxiomSet([...axiomSet, newAxiom]);

    setEvolutionHistory([
      ...evolutionHistory,
      {
        timestamp: new Date().toISOString(),
        trigger: 'User suggestion',
        oldState: axiomSet.length + ' axioms',
        newAxiom: proposedAxiom.proposed,
        synthesis: 'Direct user addition',
        context: proposedAxiom.rationale
      }
    ]);

    alert('Axiom added! Try reprocessing text to see the effect.');
  };

  const exampleTexts = [
    {
      name: "Hegel - Being/Nothing",
      text: "Being, pure being, without any further determination. In its indeterminate immediacy it is
      ↪  equal only to itself. It is also not unequal relatively to an other; it has no diversity
      ↪  within itself nor any with a reference outwards. Pure being is in fact nothing, and neither
      ↪  more nor less than nothing."
    },
    {
      name: "Hegel - Self-Consciousness",
      text: "Self-consciousness exists in and for itself when, and by the fact that, it so exists for
      ↪  another; that is, it exists only in being acknowledged."
    },
    {
      name: "Hegel - Master/Slave",
      text: "The master relates himself to the bondsman mediately through independent being, for that is
      ↪  precisely what keeps the bondsman in thrall; it is his chain, from which he could not in the
      ↪  struggle get away, and for that reason he proved himself to be dependent, to have his
      ↪  independence in the shape of thinghood."
    }
  ];

  return (
    <div className="min-h-screen bg-gradient-to-br from-slate-900 via-purple-900 to-slate-900 text-white
    ↪  p-6">
      <div className="max-w-6xl mx-auto">
        {/* Header */}
        <div className="mb-8">
          <h1 className="text-4xl font-bold mb-2 bg-gradient-to-r from-purple-400 to-pink-400
          ↪  bg-clip-text text-transparent">
            Dialectical Interpreter
          </h1>
          <p className="text-purple-300">
            A self-evolving PML system for philosophical text analysis
          </p>
        </div>

        {/* Main Input Section */}
        <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6 mb-6">
          <div className="mb-4">
            <label className="block text-sm font-medium mb-2">Philosophical Text</label>
            <textarea
              value={inputText}
              onChange={(e) => setInputText(e.target.value)}
```

```
            className="w-full h-40 bg-black/30 border border-purple-500/30 rounded-lg p-4 text-white placeholder-purple-300/50 focus:outline-none focus:border-purple-400 focus:ring-2 focus:ring-purple-400/20"
            placeholder="Paste a philosophical passage here (e.g., from Hegel's Phenomenology)..."
          />
        </div>

        <div className="flex flex-wrap gap-2 mb-4">
          <span className="text-sm text-purple-300">Examples:</span>
          {exampleTexts.map((ex, i) => (
            <button
              key={i}
              onClick={() => setInputText(ex.text)}
              className="text-xs bg-purple-500/20 hover:bg-purple-500/30 px-3 py-1 rounded-full transition-colors"
            >
              {ex.name}
            </button>
          ))}
        </div>

        <div className="flex gap-3">
          <button
            onClick={processText}
            disabled={!inputText || isProcessing}
            className="flex-1 bg-gradient-to-r from-purple-500 to-pink-500 hover:from-purple-600 hover:to-pink-600 disabled:from-gray-500 disabled:to-gray-600 px-6 py-3 rounded-lg font-medium transition-all flex items-center justify-center gap-2"
          >
            {isProcessing ? (
              <>
                <Loader2 className="w-5 h-5 animate-spin" />
                Processing...
              </>
            ) : (
              <>
                <Sparkles className="w-5 h-5" />
                {iterationDepth > 0 && inputText === interpretation?.original
                  ? `Re-read (Iteration ${iterationDepth + 1})`
                  : 'Interpret Text'}
              </>
            )}
          </button>

          <button
            onClick={() => setShowEvolutionPanel(!showEvolutionPanel)}
            className="bg-purple-500/20 hover:bg-purple-500/30 px-6 py-3 rounded-lg font-medium transition-colors flex items-center gap-2"
          >
            <GitBranch className="w-5 h-5" />
            Logic ({axiomSet.length})
          </button>
        </div>

        {/* Iteration depth indicator */}
        {iterationDepth > 0 && (
          <div className="mt-3 bg-blue-500/20 border border-blue-400/30 rounded-lg p-3">
            <p className="text-sm text-blue-200">
              <strong>Iteration {iterationDepth}</strong> -
              {formalizedConcepts.length > 0
                ? ` ${formalizedConcepts.length} concepts formalized as structural scaffolding`
                : ' First reading - all material inference'}
            </p>
            {formalizedConcepts.length > 0 && (
              <p className="text-xs text-blue-300 mt-1">
```

```
                Formalized: {formalizedConcepts.join(', ')}
              </p>
            )}
          </div>
        )}
      </div>

      {/* Evolution Panel */}
      {showEvolutionPanel && (
        <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6 mb-6">
          <div className="flex items-center justify-between mb-4">
            <h2 className="text-xl font-bold flex items-center gap-2">
              <GitBranch className="w-5 h-5" />
              Axiom Evolution
            </h2>
            <button
              onClick={exportAxiomsAsProlog}
              className="bg-blue-500/30 hover:bg-blue-500/40 px-4 py-2 rounded text-sm
              ↪  transition-colors"
            >
               Export as Prolog
            </button>
          </div>

          <div className="space-y-3 mb-6">
            {axiomSet.map((axiom) => (
              <div
                key={axiom.id}
                className={`p-4 rounded-lg ${
                  axiom.source === 'core'
                    ? 'bg-blue-500/20 border border-blue-500/30'
                    : 'bg-green-500/20 border border-green-500/30'
                }`}
              >
                <div className="flex items-start justify-between">
                  <div className="flex-1">
                    <div className="flex items-center gap-2 mb-2">
                      <code className="text-sm font-mono text-purple-200">{axiom.content}</code>
                    </div>
                    <p className="text-xs text-purple-300 mb-1">{axiom.context}</p>
                    {axiom.rationale && (
                      <p className="text-xs text-purple-300 mb-1">Rationale: {axiom.rationale}</p>
                    )}
                    {axiom.addresses && (
                      <p className="text-xs text-green-300 mt-1">Addresses: {axiom.addresses}</p>
                    )}
                  </div>
                  <div className="flex flex-col gap-1 ml-3">
                    <span className={`text-xs px-2 py-1 rounded ${
                      axiom.source === 'core' ? 'bg-blue-500/30' : 'bg-green-500/30'
                    }`}>
                      {axiom.source}
                    </span>
                    <span className={`text-xs px-2 py-1 rounded ${
                      axiom.type === 'formal' ? 'bg-yellow-500/30' : 'bg-purple-500/30'
                    }`}>
                      {axiom.type}
                    </span>
                  </div>
                </div>
              </div>
            ))}
          </div>

          {evolutionHistory.length > 0 && (
```

```
            <>
              <h3 className="text-lg font-bold mb-3">Evolution History</h3>
              <div className="space-y-2">
                {evolutionHistory.map((event, i) => (
                  <div key={i} className="bg-black/30 p-3 rounded-lg text-sm">
                    <div className="flex items-center gap-2 mb-1">
                      <CheckCircle className="w-4 h-4 text-green-400" />
                      <span className="text-purple-300">{new
                      ↪  Date(event.timestamp).toLocaleTimeString()}</span>
                    </div>
                    <p className="text-yellow-300 mb-1">Trigger: {event.trigger}</p>
                    <p className="text-green-300">New Axiom: <code>{event.newAxiom}</code></p>
                    <p className="text-purple-200 text-xs mt-1">{event.synthesis}</p>
                  </div>
                ))}
              </div>
            </>
          )}
        </div>
      )}

      {/* Results Section */}
      {interpretation && !interpretation.error && (
        <div className="space-y-6">
          {/* Export Controls */}
          <div className="bg-gradient-to-r from-green-500/20 to-blue-500/20 backdrop-blur-lg
          ↪  rounded-lg p-4 border border-green-400/30">
            <div className="flex items-center justify-between">
              <div>
                <h3 className="font-semibold text-green-300 mb-1">Export Analysis</h3>
                <p className="text-xs text-green-200">
                  Copy interpretation or export evolved axioms as Prolog code
                </p>
              </div>
              <div className="flex gap-2">
                <button
                  onClick={copyInterpretation}
                  className="bg-green-500/30 hover:bg-green-500/40 px-4 py-2 rounded transition-colors
                  ↪  text-sm"
                >
                   Copy Reading
                </button>
                <button
                  onClick={exportAxiomsAsProlog}
                  className="bg-blue-500/30 hover:bg-blue-500/40 px-4 py-2 rounded transition-colors
                  ↪  text-sm"
                >
                   Export Prolog
                </button>
              </div>
            </div>
          </div>
          {/* PML Formalization */}
          <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6">
            <div className="mb-4">
              <h2 className="text-xl font-bold mb-2">Phenomenological Reading (PML)</h2>
              <p className="text-purple-300 text-sm italic">
                Tracking the temporal, embodied experience of working through this text ↓↑
              </p>
              {interpretation.pml.reading_experience && (
                <div className="mt-3 bg-purple-500/20 p-3 rounded-lg">
                  <p className="text-sm text-purple-100">{interpretation.pml.reading_experience}</p>
                </div>
              )}
            </div>
```

```
746
747             <div className="space-y-3">
748               {interpretation.pml.formalizations.map((form, i) => (
749                 <div key={i} className="bg-black/30 p-4 rounded-lg">
750                   <div className="flex items-center gap-2 mb-2">
751                     <span className="bg-purple-500/30 px-2 py-1 rounded text-xs font-medium">
752                       {form.step}
753                     </span>
754                     {form.temporal_moment && (
755                       <span className="bg-blue-500/30 px-2 py-1 rounded text-xs">
756                         {form.temporal_moment}
757                       </span>
758                     )}
759                     {form.inference_type && (
760                       <span className={`px-2 py-1 rounded text-xs ${
761                         form.inference_type === 'formal'
762                           ? 'bg-yellow-500/30 border border-yellow-400/30'
763                           : 'bg-green-500/30'
764                       }`}>
765                         {form.inference_type === 'formal' ? '  formal' : '  material'}
766                       </span>
767                     )}
768                   </div>
769                   <code className="text-purple-300 block mb-2">{form.pml}</code>
770                   <p className="text-sm text-purple-200">{form.explanation}</p>
771                 </div>
772               ))}
773             </div>
774           </div>
775
776           {/* Logical Proof */}
777           <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6">
778             <h2 className="text-xl font-bold mb-4">Proof Steps</h2>
779             <div className="space-y-2">
780               {interpretation.logic.proof_steps.map((step, i) => (
781                 <div key={i} className="bg-black/30 p-3 rounded-lg">
782                   <div className="flex items-start gap-3">
783                     <span className="bg-blue-500/30 px-2 py-1 rounded text-xs font-mono shrink-0">
784                       {i + 1}
785                     </span>
786                     <div className="flex-1">
787                       <p className="text-sm text-blue-300 mb-1">
788                         Premises: {step.premises.join(', ')}
789                       </p>
790                       <p className="text-sm text-purple-300 mb-1">
791                         Axiom: <code>{step.axiom_used}</code>
792                       </p>
793                       <p className="text-sm text-green-300 mb-1">
794                         → Conclusion: <code>{step.conclusion}</code>
795                       </p>
796                       <p className="text-xs text-purple-200">{step.explanation}</p>
797                     </div>
798                   </div>
799                 </div>
800               ))}
801             </div>
802           </div>
803
804           {/* Interpretation */}
805           <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6">
806             <h2 className="text-xl font-bold mb-4">Interpretation</h2>
807             <p className="text-purple-100 mb-4
      ↪  leading-relaxed">{interpretation.logic.interpretation}</p>
808
809             {interpretation.logic.key_insights && interpretation.logic.key_insights.length > 0 && (
```

```
              <div className="mt-4">
                <h3 className="font-semibold mb-2">Key Insights:</h3>
                <ul className="space-y-1">
                  {interpretation.logic.key_insights.map((insight, i) => (
                    <li key={i} className="text-sm text-purple-200 flex items-start gap-2">
                      <CheckCircle className="w-4 h-4 text-green-400 mt-0.5 shrink-0" />
                      {insight}
                    </li>
                  ))}
                </ul>
              </div>
            )}
          </div>

          {/* Critique & Evolution */}
          <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6">
            <h2 className="text-xl font-bold mb-4 flex items-center gap-2">
              <AlertCircle className="w-5 h-5" />
              Meta-Critique: Levels of Analysis
            </h2>

            {/* Meta-Insight */}
            {interpretation.critique.diagnostic.meta_insight && (
              <div className="mb-6 bg-gradient-to-r from-purple-500/20 to-pink-500/20 p-4 rounded-lg
              ↪  border border-purple-400/30">
                <h3 className="font-semibold mb-2 text-purple-300">🔄 Meta-Insight:</h3>
                <p className="text-sm
                ↪  text-purple-100">{interpretation.critique.diagnostic.meta_insight}</p>
              </div>
            )}

            {/* Established Readings */}
            <div className="mb-6">
              <h3 className="font-semibold mb-3">Established Scholarly Readings:</h3>
              <div className="space-y-2">
                {interpretation.critique.established_readings.map((reading, i) => (
                  <div key={i} className="bg-blue-500/20 p-3 rounded-lg">
                    <div className="flex items-center gap-2 mb-1">
                      <p className="text-sm font-medium text-blue-300">{reading.scholar}</p>
                      {reading.level && (
                        <span className="text-xs bg-blue-500/30 px-2 py-0.5 rounded">
                          {reading.level}
                        </span>
                      )}
                    </div>
                    <p className="text-sm text-blue-200">{reading.interpretation}</p>
                  </div>
                ))}
              </div>
            </div>

            {/* Level Distinctions */}
            {interpretation.critique.alignment.level_distinctions &&
             interpretation.critique.alignment.level_distinctions.length > 0 && (
              <div className="mb-6">
                <h3 className="font-semibold mb-3 text-green-400">
                  ✓ Level Distinctions (Not Contradictions):
                </h3>
                <div className="space-y-2">
                  {interpretation.critique.alignment.level_distinctions.map((dist, i) => (
                    <div key={i} className="bg-green-500/20 p-3 rounded-lg border
                    ↪  border-green-500/30">
                      <p className="text-sm text-yellow-300 mb-1">
                        Apparent: {dist.apparent_contradiction}
                      </p>
```

```
                    <p className="text-sm text-green-300 mb-1">
                      Actually: {dist.actually}
                    </p>
                    <p className="text-xs text-green-200">{dist.explanation}</p>
                  </div>
                ))}
              </div>
            </div>
          )}

          {/* Genuine Contradictions */}
          {interpretation.critique.alignment.genuine_contradictions &&
           interpretation.critique.alignment.genuine_contradictions.length > 0 && (
            <div className="mb-6">
              <p className="text-sm text-red-400 mb-2">⚠ Genuine Contradictions (Same Level):</p>
              <div className="space-y-3">
                {interpretation.critique.alignment.genuine_contradictions.map((contra, i) => (
                  <div
                    key={i}
                    className={`p-4 rounded-lg ${
                      contra.severity === 'high' ? 'bg-red-500/20 border border-red-500/30' :
                      contra.severity === 'medium' ? 'bg-yellow-500/20 border border-yellow-500/30'
                      ↪  :
                      'bg-orange-500/20 border border-orange-500/30'
                    }`}
                  >
                    <p className="font-medium mb-2">{contra.issue}</p>
                    <p className="text-sm text-red-200 mb-1">Our claim: {contra.our_claim}</p>
                    <p className="text-sm text-yellow-200 mb-3">Standard:
                    ↪  {contra.standard_claim}</p>

                    {interpretation.critique.diagnostic.needed_axioms
                      .filter(ax => ax.addresses === contra.issue)
                      .map((axiom, j) => (
                        <div key={j} className="mt-3 bg-black/30 p-3 rounded">
                          <p className="text-sm text-purple-300 mb-2">Proposed Resolution:</p>
                          <code className="text-xs text-green-300 block
                          ↪  mb-2">{axiom.proposed}</code>
                          <p className="text-xs text-purple-200 mb-3">{axiom.rationale}</p>
                          <div className="flex gap-2">
                            <button
                              onClick={() => accommodateContradiction(contra, axiom)}
                              disabled={isProcessing}
                              className="bg-green-500/30 hover:bg-green-500/40 px-4 py-2 rounded
                              ↪  text-sm transition-colors disabled:opacity-50"
                            >
                              Refine & Evolve
                            </button>
                            <button
                              onClick={() => autoAddAxiom(axiom)}
                              disabled={isProcessing}
                              className="bg-blue-500/30 hover:bg-blue-500/40 px-4 py-2 rounded
                              ↪  text-sm transition-colors disabled:opacity-50"
                            >
                              ⚡ Quick Add
                            </button>
                          </div>
                        </div>
                      ))}
                  </div>
                ))}
              </div>
            </div>
          )}

```

```
            {/* Pathology Detection */}
            {interpretation.critique.diagnostic.pathology_detected !== 'none' && (
              <div className="bg-red-500/20 border border-red-500/30 p-4 rounded-lg">
                <h3 className="font-semibold mb-2 flex items-center gap-2">
                  <XCircle className="w-5 h-5" />
                  Pathology Detected: {interpretation.critique.diagnostic.pathology_detected}
                </h3>
                <p className="text-sm text-red-200">
                  The current axiom set may be generating a pathological pattern.
                  Consider accepting the proposed axioms to achieve sublation.
                </p>
              </div>
            )}
          </div>

          {/* Follow-up Conversation */}
          {showFollowUp && (
            <div className="bg-white/10 backdrop-blur-lg rounded-lg p-6">
              <h2 className="text-xl font-bold mb-4">Dialectical Refinement</h2>
              <p className="text-purple-300 text-sm mb-4">
                Clarify the interpretation, ask about specific moves, or challenge the framing
              </p>

              {/* Previous follow-ups */}
              {interpretation.followUps && interpretation.followUps.length > 0 && (
                <div className="mb-4 space-y-3">
                  {interpretation.followUps.map((fu, i) => (
                    <div key={i} className="space-y-2">
                      <div className="bg-blue-500/20 p-3 rounded-lg">
                        <p className="text-sm font-medium text-blue-300 mb-1">You asked:</p>
                        <p className="text-sm text-blue-100">{fu.question}</p>
                      </div>
                      <div className="bg-purple-500/20 p-3 rounded-lg">
                        <p className="text-sm font-medium text-purple-300 mb-1">Response:</p>
                        <p className="text-sm text-purple-100 whitespace-pre-wrap">{fu.response}</p>
                      </div>
                    </div>
                  ))}
                </div>
              )}

              {/* New follow-up input */}
              <div className="flex gap-3">
                <textarea
                  value={followUpQuestion}
                  onChange={(e) => setFollowUpQuestion(e.target.value)}
                  placeholder="e.g., 'But isn't the temporal reading inconsistent with Hegel's claim
                  ↪  that logic is atemporal?' or 'What about the role of negation here?'"
                  className="flex-1 bg-black/30 border border-purple-500/30 rounded-lg p-3 text-white
                  ↪  placeholder-purple-300/50 focus:outline-none focus:border-purple-400
                  ↪  focus:ring-2 focus:ring-purple-400/20 min-h-[80px]"
                  disabled={isProcessing}
                />
                <button
                  onClick={handleFollowUp}
                  disabled={!followUpQuestion.trim() || isProcessing}
                  className="bg-gradient-to-r from-purple-500 to-pink-500 hover:from-purple-600
                  ↪  hover:to-pink-600 disabled:from-gray-500 disabled:to-gray-600 px-6 rounded-lg
                  ↪  font-medium transition-all self-end"
                >
                  {isProcessing ? <Loader2 className="w-5 h-5 animate-spin" /> : 'Ask'}
                </button>
              </div>
            </div>
          )}
```

```
          </div>
        )}

        {interpretation && interpretation.error && (
          <div className="bg-red-500/20 border border-red-500/30 rounded-lg p-6">
            <h2 className="text-xl font-bold mb-2 flex items-center gap-2">
              <XCircle className="w-5 h-5" />
              Error
            </h2>
            <p className="text-red-200">{interpretation.message}</p>
          </div>
        )}
      </div>
    </div>
  );
};

export default DialecticalInterpreter;

```

## 2 src/main.jsx

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import DialecticalInterpreter from './DialecticalInterpreter';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <DialecticalInterpreter />
  </React.StrictMode>
);

```