自反计算系统开发者版

```
<!DOCTYPE html>
<!-- saved from url=
(0122)file:///C:/Users/lzy/AppData/Local/Temp/CherryStudio/temp_file_c6639f0
1-bae2-450f-9adc-db7432ade27d_artifacts-preview.html -->
<html lang="zh-CN"><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TermCraft: 形式系统解谜游戏</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <style>
        /* 自定义样式，用于增强交互性 */
        .term-node {
            transition: background-color 0.2s ease-in-out, box-shadow 0.2s
ease-in-out;
            cursor: default;
        }
        .term-node.highlight-app, .term-node.highlight-rule-l, .term-
node.highlight-rule-r {
            box-shadow: 0 0 0 2px;
        }
        .term-node.highlight-app { box-shadow-color: #3b82f6; /* blue-500 */
}
        .term-node.highlight-rule-l { box-shadow-color: #ef4444; /* red-500
*/ }
        .term-node.highlight-rule-r { box-shadow-color: #22c55e; /* green-
500 */ }
        .term-node.highlight-placeholder { background-color: #eab308; /*
yellow-500 */ color: #fff; cursor: pointer; }
        .term-node.highlight-placeholder:hover { background-color: #f59e0b;
}

        .action-card {
            transition: transform 0.1s ease-in-out, box-shadow 0.2s ease-in-
out;
        }
        .action-card:hover {
            transform: translateY(-2px);
            box-shadow: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px
rgb(0 0 0 / 0.1);
        }
    </style>
</head>
<body class="bg-slate-100 font-sans text-slate-800 p-4 lg:p-8">
```

```
    <div id="game-container" class="max-w-7xl mx-auto">
        <header class="mb-6 pb-4 border-b border-slate-300">
            <div class="flex items-center justify-between">
                <div>
                    <h1 class="text-4xl font-bold text-slate-
900">TermCraft</h1>
                    <p class="text-slate-600 mt-1">一个基于形式系统重写的解谜游
戏引擎</p>
                </div>
                <button id="dev-toggle-button" class="bg-slate-200 text-
slate-800 font-semibold py-2 px-4 rounded-lg hover:bg-slate-300 transition-
colors">
                    开发者模式
                </button>
            </div>
        </header>

        <!-- 开发者模式面板 -->
        <div id="dev-panel" class="hidden bg-white p-6 rounded-lg shadow-md
mb-6 space-y-4">
            <h2 class="text-2xl font-semibold text-slate-800 border-b pb-2">
开发者设置</h2>
            <div class="grid grid-cols-1 lg:grid-cols-2 gap-4">
                <div>
                    <label for="dev-fuel-input" class="block text-sm font-
semibold text-slate-700 mb-1">燃料 (Fuel)</label>
                    <input type="number" id="dev-fuel-input" class="w-full
border border-slate-300 rounded-md p-2">
                </div>
                <div>
                    <label for="dev-budget-input" class="block text-sm font-
semibold text-slate-700 mb-1">预算 (Budget)</label>
                    <input type="number" id="dev-budget-input" class="w-full
border border-slate-300 rounded-md p-2">
                </div>
            </div>
            <div>
                <label for="dev-current-term-input" class="block text-sm
font-semibold text-slate-700 mb-1">当前项 (Current Term)</label>
                <textarea id="dev-current-term-input" rows="5" class="w-full
border border-slate-300 rounded-md p-2 font-mono text-sm"></textarea>
            </div>
            <div>
                <label for="dev-target-term-input" class="block text-sm
font-semibold text-slate-700 mb-1">目标项 (Target Term)</label>
                <textarea id="dev-target-term-input" rows="5" class="w-full
border border-slate-300 rounded-md p-2 font-mono text-sm"></textarea>
            </div>
            <div class="flex justify-end items-center gap-4">
```

```
                <p id="dev-error-message" class="text-red-500 text-sm flex-grow"></p>
                <button id="dev-apply-button" class="bg-indigo-600 text-white font-semibold py-2 px-6 rounded-lg hover:bg-indigo-700 transition-colors">
                    应用设置
                </button>
            </div>
        </div>

        <!-- 游戏状态显示 -->
        <div id="game-status" class="flex flex-wrap gap-4 mb-6 text-center">
            <div class="bg-blue-100 text-blue-800 p-4 rounded-lg shadow-sm flex-grow">
                <div class="text-sm font-semibold uppercase">剩余燃料 (Fuel)</div>
                <div id="fuel-display" class="text-3xl font-bold">5</div>
            </div>
            <div class="bg-green-100 text-green-800 p-4 rounded-lg shadow-sm flex-grow">
                <div class="text-sm font-semibold uppercase">剩余预算 (Budget)</div>
                <div id="budget-display" class="text-3xl font-bold">10</div>
            </div>
        </div>

        <!-- 游戏主面板 -->
        <main class="grid grid-cols-1 lg:grid-cols-2 gap-8">
            <!-- 左侧：当前项和目标项 -->
            <div>
                <section id="current-term-section" class="bg-white p-6 rounded-lg shadow-md mb-6">
                    <h2 class="text-2xl font-semibold mb-3 text-slate-800">当前项 (Current Term)</h2>
                    <div id="current-term-display" class="bg-slate-50 p-4 rounded-md text-lg font-mono overflow-x-auto"></div>
                </section>

                <section id="target-term-section" class="bg-white p-6 rounded-lg shadow-md">
                    <h2 class="text-2xl font-semibold mb-3 text-slate-800">目标项 (Target Term)</h2>
                    <div id="target-term-display" class="bg-slate-200 p-4 rounded-md text-lg font-mono overflow-x-auto"></div>
                </section>
            </div>

            <!-- 右侧：可用动作 -->
            <div>
                <section id="actions-section" class="bg-white p-6 rounded-lg
```

```
shadow-md">
                    <h2 class="text-2xl font-semibold mb-3 text-slate-800">
可用动作</h2>
                    <div id="actions-list" class="space-y-4 max-h-[60vh]
overflow-y-auto pr-2"></div>
                </section>
            </div>
        </main>

        <!-- 游戏消息模态框（胜利/失败） -->
        <div id="message-modal" class="fixed inset-0 bg-black bg-opacity-50
items-center justify-center flex hidden">
            <div id="message-content" class="bg-white text-center p-8
rounded-lg shadow-2xl max-w-sm">
                <h3 id="message-title" class="text-4xl font-bold mb-4">胜利!
</h3>
                <p id="message-body" class="text-slate-600 mb-6">恭喜你，成功
构造出目标项！</p>
                <button id="restart-button" class="bg-indigo-600 text-white
font-semibold py-2 px-6 rounded-lg hover:bg-indigo-700 transition-colors">重
新开始</button>
            </div>
        </div>

         <!-- 构造模态框 -->
        <div id="construct-modal" class="fixed inset-0 bg-black bg-opacity-
50 items-center justify-center hidden">
            <div class="bg-white p-8 rounded-lg shadow-2xl w-full max-w-md">
                <h3 class="text-2xl font-bold mb-4 text-slate-800">构造新项
</h3>
                <p class="mb-4 text-slate-600">输入一个新项来填充占位符。大小不能
超过预算 <span id="construct-budget" class="font-bold">10</span>。</p>
                <textarea id="construct-input" rows="3" class="w-full border
border-slate-300 rounded-md p-2 font-mono focus:ring-2 focus:ring-indigo-500
focus:border-indigo-500"></textarea>
                <p id="construct-error" class="text-red-500 text-sm mt-1 h-
5"></p>
                <div class="mt-4 flex justify-end gap-3">
                    <button id="construct-cancel" class="bg-slate-200 text-
slate-800 font-semibold py-2 px-4 rounded-lg hover:bg-slate-300 transition-
colors">取消</button>
                    <button id="construct-confirm" class="bg-indigo-600
text-white font-semibold py-2 px-4 rounded-lg hover:bg-indigo-700
transition-colors">确认构造</button>
                </div>
            </div>
        </div>

    </div>
```

```
<script>
//
================================================================================
// TermCraft: 形式系统解谜游戏引擎 - JavaScript 实现
//
================================================================================

// ------------------------------------------------------------------------
---
// Section 1: 数据结构与游戏状态
// ------------------------------------------------------------------------
---

/**
 * Term: 代表形式系统中的项结构（树状）。
 * type: 'VAR', 'CONST', 'COMPOUND'
 * value: 变量名, 常量名, 或复合项的构造函数名
 * children: 子项 Term 对象的数组
 */
class Term {
    constructor(type, value, children = []) {
        this.type = type;
        this.value = value;
        this.children = children;
        this.id = Math.random().toString(36).substr(2, 9); // 用于 DOM 元素的
唯一 ID
    }

    // 工厂方法
    static Var(name) { return new Term('VAR', name); }
    static Const(name) { return new Term('CONST', name); }
    static Compound(name, children) { return new Term('COMPOUND', name,
children); }

    isVariable() { return this.type === 'VAR'; }
    isConstant() { return this.type === 'CONST'; }
    isCompound() { return this.type === 'COMPOUND'; }

    getConstructor() { return this.value; }
    getArity() { return this.children.length; }
    getChild(i) { return this.children[i]; }

    // 用于比较和作为 Map 的 key
    toString() {
        if (this.isVariable()) return `var(${this.value})`;
        if (this.isConstant()) return this.value;
        return `${this.value}(${this.children.map(c =>
c.toString()).join(',')})`;
    }
```

```
    // 深拷贝
    clone() {
        return new Term(this.type, this.value, this.children.map(c =>
c.clone()));
    }

    // 深比较
    equals(other) {
        if (!(other instanceof Term)) return false;
        return this.toString() === other.toString();
    }

    // 用于解析字符串输入
    static fromString(str) {
        str = str.trim();
        const varMatch = str.match(/^var\((\w+)\)$/i); // Case-insensitive
match for var
        if (varMatch) {
            return Term.Var(varMatch[1]);
        }
        const compoundMatch = str.match(/^(\w+)\((.*)\)$/);
        if (compoundMatch) {
            const name = compoundMatch[1];
            const argsStr = compoundMatch[2];
            if (argsStr.trim() === '') return Term.Compound(name, []);

            const children = [];
            let balance = 0;
            let lastSplit = 0;
            for (let i = 0; i < argsStr.length; i++) {
                if (argsStr[i] === '(') balance++;
                else if (argsStr[i] === ')') balance--;
                else if (argsStr[i] === ',' && balance === 0) {

children.push(Term.fromString(argsStr.substring(lastSplit, i)));
                    lastSplit = i + 1;
                }
            }
            children.push(Term.fromString(argsStr.substring(lastSplit)));
            return Term.Compound(name, children);
        }
        if (str.match(/^\w+$/)) {
            return Term.Const(str);
        }
        throw new Error(`Invalid term string: "${str}"`);
    }
}

/**
 * UnionFind: 用于高效处理等价关系。
```

```
 */
class UnionFind {
    constructor(elements) {
        this.parent = new Map();
        elements.forEach(el => this.parent.set(el.toString(), el));
    }

    find(tStr) {
        if (this.parent.get(tStr).toString() === tStr) {
            return this.parent.get(tStr);
        }
        const root = this.find(this.parent.get(tStr).toString());
        this.parent.set(tStr, root); // Path compression
        return root;
    }

    union(t1, t2) {
        const root1 = this.find(t1.toString());
        const root2 = this.find(t2.toString());
        if (!root1.equals(root2)) {
            this.parent.set(root2.toString(), root1);
        }
    }

    getClass(t) {
        const rootT = this.find(t.toString());
        const result = new Set();
        for (const [key, _] of this.parent.entries()) {
            if (this.find(key).equals(rootT)) {
                 result.add(Term.fromString(key));
            }
        }
        return Array.from(result);
    }
}

// ----------------------------------------------------------------------
---
// Section 2: 主游戏循环 (由UI事件驱动)
// ----------------------------------------------------------------------
---

let GameState = { T: null, F: 0, B: 0 };
let TargetTerm = null;
let InitialState = {};

function MainGameLoop(T_initial, F_initial, B_initial, T_target) {
    // 1. 初始化游戏状态
    GameState = {
        T: T_initial.clone(),
```

```
        F: F_initial,
        B: B_initial
    };
    TargetTerm = T_target.clone();
    InitialState = { T: T_initial.clone(), F: F_initial, B: B_initial,
Target: T_target.clone() };

    runTurn();
}

function runTurn() {
    // 2. 向玩家展示当前状态
    DisplayToPlayer(GameState.T, GameState.F, GameState.B, TargetTerm);

    // 3. 检查胜利条件
    if (GameState.T.equals(TargetTerm)) {
        DeclareVictory();
        return;
    }

    // 4. 检查失败条件
    if (GameState.F <= 0) {
        DeclareDefeat("燃料耗尽！");
        return;
    }

    // 5. 计算当前回合的背景理论
    const equiv_relation = CalculateEquivalenceRelation(GameState.T);

    // 6. 找出所有合法的玩家动作
    const valid_rewrites = FindValidRewrites(GameState.T, equiv_relation);
    const placeholder_sites = FindPlaceholderSites(GameState.T);

    // 7. 检查是否陷入死局
    if (valid_rewrites.length === 0 && placeholder_sites.length === 0) {
        DeclareDefeat("没有更多可用动作！");
        return;
    }

    // 8. 将动作展示给玩家，等待玩家选择 (GetPlayerChoice)
    displayActions(valid_rewrites, placeholder_sites);
}

// ------------------------------------------------------------------------
---
// Section 3: 核心算法
// ------------------------------------------------------------------------
---

function CalculateEquivalenceRelation(T) {
```

```
    const ground_subterms = GetAllGroundSubterms(T);
    const union_find = new UnionFind(ground_subterms);
    const equiv_axioms = FindAllSubterms(T, "equiv");
    for (const axiom of equiv_axioms) {
        if (axiom.getArity() === 2) {
            union_find.union(axiom.getChild(0), axiom.getChild(1));
        }
    }
    return union_find;
}

function FindValidRewrites(T, equiv_relation) {
    let valid_actions = [];
    const all_rules = FindAllSubtermsWithPositions(T, "rewrite");
    const all_ground_sites = GetAllGroundSubtermsWithPositions(T);

    for (const [p_rule, rule_term] of all_rules) {
        if (rule_term.getArity() !== 2) continue;
        const [l, r] = rule_term.children;

        for (const [p_app, site_term] of all_ground_sites) {
            const found_substitutions = FindInferentialMatches(l, site_term,
equiv_relation);
            for (const sigma of found_substitutions) {
                valid_actions.push({ type: 'Rewrite', p_app, p_rule, sigma
});
            }
        }
    }
    return valid_actions;
}

function FindInferentialMatches(pattern_l, term_a, equiv_relation) {
    const all_found_substitutions = [];
    const equivalence_class_of_a = equiv_relation.getClass(term_a);

    for (const b of equivalence_class_of_a) {
        const [was_successful, final_substitution] =
SyntacticMatch(pattern_l, b, new Map());
        if (was_successful) {
            all_found_substitutions.push(final_substitution);
        }
    }

    return UniqueSubstitutions(all_found_substitutions);
}

function SyntacticMatch(pattern, term, current_substitution) {
    if (pattern.isVariable()) {
        const v_str = pattern.toString();
```

```
        if (current_substitution.has(v_str)) {
            const is_consistent =
current_substitution.get(v_str).equals(term);
            return [is_consistent, current_substitution];
        } else {
            const new_substitution = new Map(current_substitution);
            new_substitution.set(v_str, term);
            return [true, new_substitution];
        }
    }

    if (pattern.isConstant()) {
        return [pattern.equals(term), current_substitution];
    }

    if (pattern.isCompound()) {
        if (!term.isCompound() || pattern.getConstructor() !==
term.getConstructor() || pattern.getArity() !== term.getArity()) {
            return [false, null];
        }

        let temp_substitution = current_substitution;
        for (let i = 0; i < pattern.getArity(); i++) {
            const [was_successful, updated_substitution] = SyntacticMatch(
                pattern.getChild(i),
                term.getChild(i),
                temp_substitution
            );
            if (!was_successful) {
                return [false, null];
            }
            temp_substitution = updated_substitution;
        }
        return [true, temp_substitution];
    }

    return [false, null]; // Should not be reached
}


function ApplyAction(action) {
    let T_next, F_next, B_next;

    if (action.type === 'Rewrite') {
        const { p_app, p_rule, sigma } = action;
        const rule_term = getSubtermAt(GameState.T, p_rule);
        const r = rule_term.getChild(1);

        const term_to_insert = ApplySubstitution(r, sigma);
        T_next = replaceSubtermAt(GameState.T, p_app, term_to_insert);
```

```
        F_next = GameState.F - 1;
        B_next = GameState.B;
    } else if (action.type === 'Construct') {
        const { p_fill, t_new } = action;
        const cost = Size(t_new);

        if (cost > GameState.B) {
            console.error(" presupuesto insuficiente- UI debería haber
evitado esto");
            return; // No change
        }

        T_next = replaceSubtermAt(GameState.T, p_fill, t_new);
        F_next = GameState.F;
        B_next = GameState.B - cost;
    }

    GameState = { T: T_next, F: F_next, B: B_next };
    runTurn();
}


// ------------------------------------------------------------------------
---
// Section 4 & 5: 辅助函数
// ------------------------------------------------------------------------
---

function TraverseWithPositions(T, visit_function, current_path = [],
accumulator = []) {
    visit_function(T, current_path, accumulator);
    if (T.isCompound()) {
        T.children.forEach((child, i) => {
            TraverseWithPositions(child, visit_function, [...current_path,
i], accumulator);
        });
    }
    return accumulator;
}

function GetAllGroundSubterms(T) {
    const visit_function = (subterm, path, acc) => {
         if (!ContainsVariables(subterm)) acc.push(subterm);
    };
    return TraverseWithPositions(T, visit_function);
}

function GetAllGroundSubtermsWithPositions(T) {
    const visit_function = (subterm, path, acc) => {
```

```
        if (!ContainsVariables(subterm)) acc.push([path, subterm]);
    };
    return TraverseWithPositions(T, visit_function);
}

function FindAllSubterms(T, constructor_name) {
    const visit_function = (subterm, path, acc) => {
        if (!subterm.isVariable() && subterm.getConstructor() ===
constructor_name) {
            acc.push(subterm);
        }
    };
    return TraverseWithPositions(T, visit_function);
}

function FindAllSubtermsWithPositions(T, constructor_name) {
    const visit_function = (subterm, path, acc) => {
        if (!subterm.isVariable() && subterm.getConstructor() ===
constructor_name) {
            acc.push([path, subterm]);
        }
    };
    return TraverseWithPositions(T, visit_function);
}

function FindPlaceholderSites(T) {
    const visit_function = (subterm, path, acc) => {
        if (!subterm.isVariable() && subterm.getConstructor() ===
"placeholder") {
            acc.push(path);
        }
    };
    return TraverseWithPositions(T, visit_function);
}

function ContainsVariables(T) {
    if (T.isVariable()) return true;
    if (T.isCompound()) {
        return T.children.some(ContainsVariables);
    }
    return false;
}

function ApplySubstitution(T, sigma) {
    if (T.isVariable()) {
        return sigma.get(T.toString()) || T;
    }
    if (T.isConstant()) {
        return T.clone();
    }
```

```
    if (T.isCompound()) {
        const new_children = T.children.map(child =>
ApplySubstitution(child, sigma));
        return Term.Compound(T.getConstructor(), new_children);
    }
}

function getSubtermAt(T, path) {
    let current = T;
    for (const index of path) {
        current = current.getChild(index);
    }
    return current;
}

function replaceSubtermAt(T, path, new_subterm) {
    if (path.length === 0) {
        return new_subterm.clone();
    }

    const root = T.clone();
    let current = root;
    for (let i = 0; i < path.length - 1; i++) {
        current = current.getChild(path[i]);
    }
    current.children[path[path.length - 1]] = new_subterm.clone();
    return root;
}

function Size(T) {
    let size = 1;
    if (T.isCompound()) {
        size += T.children.reduce((acc, child) => acc + Size(child), 0);
    }
    return size;
}

function UniqueSubstitutions(substitutions) {
    const seen = new Set();
    const unique = [];
    for (const sub of substitutions) {
        // 将 Map 转换为可比较的字符串
        const key = JSON.stringify(Array.from(sub.entries()).map(([k, v]) =>
[k, v.toString()]).sort());
        if (!seen.has(key)) {
            seen.add(key);
            unique.push(sub);
        }
    }
    return unique;
```

```
}

// ----------------------------------------------------------------------
---
// Section 5.4: UI 和游戏逻辑函数
// ----------------------------------------------------------------------
---

function termToHtml(term, path = []) {
    const pathStr = JSON.stringify(path);
    const commonClasses = 'term-node inline-block border rounded px-1 py-0.5
m-0.5';

    if (term.isVariable()) {
        return `<span class="${commonClasses} bg-purple-100 text-purple-800
border-purple-300" data-path='${pathStr}' data-id='${term.id}'>${term.value}
</span>`;
    }
    if (term.isConstant()) {
        let specialClass = 'bg-gray-100 text-gray-800 border-gray-300';
        if (term.getConstructor() === 'placeholder') {
            specialClass = 'bg-yellow-400 text-yellow-900 border-yellow-500
highlight-placeholder';
        }
        return `<span class="${commonClasses} ${specialClass}" data-
path='${pathStr}' data-id='${term.id}'>${term.value}</span>`;
    }
    if (term.isCompound()) {
        let colorClass;
        switch(term.getConstructor()) {
            case 'rewrite': colorClass = 'bg-red-50 border-red-200'; break;
            case 'equiv': colorClass = 'bg-yellow-50 border-yellow-200';
break;
            case 'tuple': colorClass = 'bg-blue-50 border-blue-200'; break;
            default: colorClass = 'bg-green-50 border-green-200';
        }
        const childrenHtml = term.children.map((child, i) =>
termToHtml(child, [...path, i])).join(', ');
        return `<span class="${commonClasses} ${colorClass}" data-
path='${pathStr}' data-id='${term.id}'><strong>${term.value}</strong>
(${childrenHtml})</span>`;
    }
}

function DisplayToPlayer(T, F, B, T_target) {
    document.getElementById('fuel-display').textContent = F;
    document.getElementById('budget-display').textContent = B;
    document.getElementById('current-term-display').innerHTML =
termToHtml(T);
    document.getElementById('target-term-display').innerHTML =
```

```
termToHtml(T_target);

    // 为 placeholders 添加事件监听器
    document.querySelectorAll('.highlight-placeholder').forEach(el => {
        el.onclick = () => {
            const path = JSON.parse(el.dataset.path);
            showConstructModal(path);
        };
    });
}

function displayActions(rewrites, placeholders) {
    const listEl = document.getElementById('actions-list');
    listEl.innerHTML = '';

    if (rewrites.length === 0 && placeholders.length === 0) {
        listEl.innerHTML = `<p class="text-slate-500 italic">没有可用的动作。
</p>`;
        return;
    }

    rewrites.forEach((action, index) => {
        const { p_app, p_rule, sigma } = action;
        const appTerm = getSubtermAt(GameState.T, p_app);
        const ruleTerm = getSubtermAt(GameState.T, p_rule);
        const [l, r] = ruleTerm.children;
        const resultTerm = ApplySubstitution(r, sigma);

        const card = document.createElement('div');
        card.className = 'action-card bg-slate-50 border border-slate-200 p-
4 rounded-lg cursor-pointer';
        card.innerHTML = `
            <div class="font-semibold text-indigo-700 mb-2">重写 #${index +
1}</div>
            <div class="font-mono text-sm space-y-2">
                <div><span class="text-slate-500">规则:</span>
rewrite(${l.toString()}, ${r.toString()})</div>
                <div><span class="text-slate-500">应用点:</span>
${appTerm.toString()}</div>
                <div><span class="text-slate-500">替换:</span>
${[...sigma.entries()].map(([k,v])=>`${k} → ${v.toString()}`).join(', ') ||
'无'}</div>
                <div class="mt-2 pt-2 border-t"><span class="text-slate-
500">结果:</span> ${appTerm.toString()} <span class="text-xl font-bold mx-2
text-indigo-500">↠</span> ${resultTerm.toString()}</div>
            </div>`;

        card.onclick = () => ApplyAction(action);

        // 高亮hover效果
```

```
        card.onmouseenter = () => highlightTerms(p_app, p_rule);
        card.onmouseleave = () => clearHighlights();

        listEl.appendChild(card);
    });
}

function highlightTerms(p_app, p_rule) {
    clearHighlights();
    const appTerm = getSubtermAt(GameState.T, p_app);
    const ruleTerm = getSubtermAt(GameState.T, p_rule);
    const lTerm = ruleTerm.getChild(0);
    const rTerm = ruleTerm.getChild(1);

    document.querySelector(`[data-
id='${appTerm.id}']`)?.classList.add('highlight-app');
    document.querySelector(`[data-
id='${lTerm.id}']`)?.classList.add('highlight-rule-l');
    document.querySelector(`[data-
id='${rTerm.id}']`)?.classList.add('highlight-rule-r');
}

function clearHighlights() {
    document.querySelectorAll('.term-node').forEach(el => {
        el.classList.remove('highlight-app', 'highlight-rule-l', 'highlight-
rule-r');
    });
}

function showConstructModal(path) {
    const modal = document.getElementById('construct-modal');
    modal.classList.remove('hidden');
    modal.classList.add('flex');

    document.getElementById('construct-budget').textContent = GameState.B;
    const input = document.getElementById('construct-input');
    input.value = '';
    input.focus();

    document.getElementById('construct-confirm').onclick = () => {
        const errorEl = document.getElementById('construct-error');
        try {
            const newTerm = Term.fromString(input.value);
            const cost = Size(newTerm);
            if (cost > GameState.B) {
                errorEl.textContent = `构造代价 (${cost}) 超过预算
(${GameState.B})!`;
                return;
            }
            errorEl.textContent = '';
```

```
            ApplyAction({ type: 'Construct', p_fill: path, t_new: newTerm
});
            hideConstructModal();
        } catch (e) {
            errorEl.textContent = "无效的项格式。示例: tuple(a,b)";
        }
    };

    document.getElementById('construct-cancel').onclick =
hideConstructModal;
}

function hideConstructModal() {
    const modal = document.getElementById('construct-modal');
    modal.classList.add('hidden');
    modal.classList.remove('flex');
}

function showMessageModal(title, body) {
    document.getElementById('message-title').textContent = title;
    document.getElementById('message-body').textContent = body;
    const modal = document.getElementById('message-modal');
    modal.classList.remove('hidden');
    modal.classList.add('flex');
}

function DeclareVictory() {
    showMessageModal("胜利!", "恭喜你，成功构造出目标项! ");
}

function DeclareDefeat(message) {
    showMessageModal("失败...", message);
}


//
================================================================================
// Section 6: 样例关卡
//
================================================================================
function loadSampleLevel() {
    const T_initial = Term.Compound("puzzle_state", [
        // Rules
        Term.Compound("rewrite", [ Term.Compound("pair", [Term.Var("X"),
Term.Var("Y")]), Term.Compound("rev_pair", [Term.Var("Y"), Term.Var("X")])
]),
        Term.Compound("rewrite", [ Term.Const("red"), Term.Const("blue") ]),
        // Axioms
        Term.Compound("equiv", [ Term.Const("apple"), Term.Const("red") ]),
        // State
```

```
        Term.Compound("data", [ Term.Compound("pair", [Term.Const("apple"),
Term.Const("green")]) ]),
        Term.Compound("data", [ Term.Const("placeholder") ])
    ]);

    const F_initial = 5;
    const B_initial = 10;

    const T_target = Term.Compound("puzzle_state", [
        Term.Compound("rewrite", [ Term.Compound("pair", [Term.Var("X"),
Term.Var("Y")]), Term.Compound("rev_pair", [Term.Var("Y"), Term.Var("X")])
]),
        Term.Compound("rewrite", [ Term.Const("red"), Term.Const("blue") ]),
        Term.Compound("equiv", [ Term.Const("apple"), Term.Const("red") ]),
        Term.Compound("data", [ Term.Compound("rev_pair",
[Term.Const("green"), Term.Const("blue")]) ]),
        Term.Compound("data", [ Term.Const("done") ])
    ]);

    MainGameLoop(T_initial, F_initial, B_initial, T_target);
}

//
================================================================================
// Section 7: 开发者工具
//
================================================================================
function populateDevPanel() {
    document.getElementById('dev-fuel-input').value = GameState.F;
    document.getElementById('dev-budget-input').value = GameState.B;
    // 使用 replace 添加换行，使其在文本框中更易读
    document.getElementById('dev-current-term-input').value =
GameState.T.toString().replace(/\),/g, '),\n');
    document.getElementById('dev-target-term-input').value =
TargetTerm.toString().replace(/\),/g, '),\n');
}

function toggleDevPanel() {
    const panel = document.getElementById('dev-panel');
    const isHidden = panel.classList.contains('hidden');
    if (isHidden) {
        populateDevPanel();
        panel.classList.remove('hidden');
    } else {
        panel.classList.add('hidden');
    }
}

function applyDevSettings() {
    const errorEl = document.getElementById('dev-error-message');
```

```
    errorEl.textContent = ''; // 清除之前的错误信息

    try {
        const fuel = parseInt(document.getElementById('dev-fuel-
input').value, 10);
        const budget = parseInt(document.getElementById('dev-budget-
input').value, 10);

        // 在解析前，移除所有空白字符（包括为可读性添加的换行符）
        const currentTermStr = document.getElementById('dev-current-term-
input').value.replace(/\s/g, '');
        const targetTermStr = document.getElementById('dev-target-term-
input').value.replace(/\s/g, '');

        if (isNaN(fuel) || isNaN(budget) || fuel < 0 || budget < 0) {
            throw new Error('燃料和预算必须是非负整数。');
        }
        if (currentTermStr === '' || targetTermStr === '') {
            throw new Error('当前项和目标项不能为空。');
        }

        const currentTerm = Term.fromString(currentTermStr);
        const targetTerm = Term.fromString(targetTermStr);

        // 如果成功，使用新设置启动游戏
        MainGameLoop(currentTerm, fuel, budget, targetTerm);
    } catch (e) {
        console.error("开发者设置错误:", e);
        errorEl.textContent = `应用失败: ${e.message}`;
    }
}

function initDevTools() {
    document.getElementById('dev-toggle-button').addEventListener('click',
toggleDevPanel);
    document.getElementById('dev-apply-button').addEventListener('click',
applyDevSettings);
}

//
================================================================================
// Section 8: 启动
//
================================================================================
document.addEventListener('DOMContentLoaded', () => {
    loadSampleLevel();
    document.getElementById('restart-button').addEventListener('click', ()
=> {
        document.getElementById('message-modal').classList.add('hidden');
        // 重新开始时加载初始关卡或上一次的自定义关卡
```

```
        MainGameLoop(InitialState.T, InitialState.F, InitialState.B,
InitialState.Target);
    });
    // 初始化开发者工具的事件监听
    initDevTools();
});

</script>

</body></html>
```

TermCraft: 形式系统解谜游戏 - 玩家与关卡设计手册

1. 游戏概述

- **一句话简介**: `TermCraft` 是一款基于形式系统（formal system）的解谜游戏，玩家通过应用重写规则和构造新结构，将一个初始的符号表达式（“项”）变换为指定的目标形态。
- **游戏目标**: 玩家的最终目标是在有限的资源（燃料和预算）内，通过一系列合法的操作，将屏幕上方的“当前项 (Current Term)”精确地变换成与“目标项 (Target Term)”完全一致的结构。

2. 核心玩法与机制

2.1 理解“项 (Term)”

游戏中的所有事物都由“项”构成。一个项是一种精确的、树状的数据结构。你可以将整个游戏世界看作一棵巨大的、不断变化的“项之树”。项有三种基本类型：

- **常量 (Constant)**: 最基础的构建块，代表一个固定的值。例如：`apple`，`green`，`placeholder`。
- **变量 (Variable)**: 在规则中作为“通配符”使用，可以匹配任意的项。格式总是 `var(NAME)`。例如：`var(X)`，`var(AnyTerm)`。
- **复合项 (Compound Term)**: 由一个构造函数名称和零个或多个子项组成，用于构建更复杂的结构。格式为 `name(child1, child2, ...)`。例如：`pair(apple, green)`。

2.2 关键资源

- **燃料 (Fuel)**: 你的主要行动力。每当你执行一次“重写”动作时，都会消耗1点燃料。燃料一旦耗尽，你将无法再执行重写，游戏可能会因此失败。
- **预算 (Budget)**: 用于创造新事物的资源。当你执行“构造”动作来填充一个占位符时，会消耗预算。消耗量等于你构造的新项的“代价”。

2.3 玩家的动作

- **动作一：重写 (Rewrite)**:
  - 游戏世界中包含形如 `rewrite(L, R)` 的规则项。`L` (Left-Hand Side) 是一个模式，`R` (Right-Hand Side) 是替换结果。

    - 当“当前项”的某个部分与规则的 `L` 部分成功匹配时，你就可以执行一次重写，将匹配部分替换为 `R`。
    - 游戏中还可能存在 `equiv(A, B)` 这样的等价公理，它会让你在匹配 `L` 时，可以将 `A` 当作 `B` 来看待（反之亦然），从而发现更多重写机会。
    - **代价**: 每次重写消耗 **1** 点燃料。
- **动作二：构造 (Construct)**:
    - 游戏世界中存在名为 `placeholder` 的特殊常量。它们在界面上高亮显示，代表可以填充新内容的地方。
    - 点击一个 `placeholder`，你可以输入一个全新的项来替换它。
    - **代价**: 构造会消耗预算。新项的“代价”不能超过你当前剩余的预算。代价的计算方式详见 3.4 节。

2.4 胜利与失败

- **胜利条件**: 当“当前项”的结构与“目标项”完全一致时，你将获得胜利。
- **失败条件**:
    - 燃料耗尽，且无法通过构造新项来继续游戏。
    - 没有任何可执行的重写或构造动作，陷入死局。

3. 开发者模式与关卡设计指南 (高级)

3.1 开启方式与面板概览

点击游戏界面右上角的 **“开发者模式”** 按钮即可开启设置面板。你可以在此定义一个全新的关卡。

字段名称	HTML ID	说明
燃料 (Fuel)	`dev-fuel-input`	关卡的初始燃料值。
预算 (Budget)	`dev-budget-input`	关卡的初始预算值。
当前项	`dev-current-term-input`	关卡的初始状态，以项的文本格式输入。
目标项	`dev-target-term-input`	关卡的胜利条件，以项的文本格式输入。
应用设置	`dev-apply-button`	点击此按钮，用以上设置加载新关卡。

3.2 “项”的通用语法规则

- **分析来源**: 以下所有语法规则均由游戏引擎的 `Term.fromString(str)` 解析函数严格定义。
- **语法细则**:
    - **常量 (Constant)**:
        - **格式**: 由字母、数字和下划线组成的单个词。
        - **规则**: 不能包含括号或逗号，除非它们是整个名称的一部分（不推荐）。

    - **示例**: `apple`, `my_const`, `state1`
- **变量 (Variable)**:
    - **格式**: `var(Name)`
    - **规则**: `var` 关键字本身不区分大小写，但括号和内部的变量名 `Name` 是必需的。 `Name` 遵循与常量相同的命名规则。
    - **示例**: `var(X)`, `var(Anything)`
- **复合项 (Compound Term)**:
    - **格式**: `name(argument1, argument2, ...)`
    - **规则**:
        - 以构造函数名称开头，后跟一对括号。
        - 括号内是零个或多个子项（参数），以逗号 `,` 分隔。
        - 子项可以是任何合法的项类型（常量、变量、其他复合项），允许无限嵌套。
        - 构造函数名称遵循与常量相同的命名规则。
        - 零参数的复合项格式为 `name()`。
    - **示例**: `pair(apple, green)`, `rule(var(X), var(X))`, `empty_list()`

3.3 核心构造函数详解

引言: 在TermCraft中，大多数复合项的名称只起到组织数据的作用。然而，有几个特殊的名称是游戏引擎的‘关键字’，它们会触发核心的游戏机制。理解它们的区别是设计关卡的关键。

3.3.1 机制驱动型构造函数 (Engine Keywords)

一、 重写规则： `rewrite`

- **结构**: `rewrite(LHS, RHS)`
- **作用**: 定义一条核心的重写规则。这是玩家可以主动选择并执行的主要动作。游戏引擎会扫描所有 `rewrite` 项来生成“可用动作”列表。
- **参数**:
    - `LHS` (Left-Hand Side): 一个作为“模式”的项。它可以包含变量。
    - `RHS` (Right-Hand Side): 一个作为“替换结果”的项。
- **机制**: 当玩家选择应用此规则，并且游戏中的某个地方（应用点）能与 `LHS` 匹配时，该应用点就会被替换为代入变量后的 `RHS`。此操作消耗1点燃料。
- **示例**: `rewrite(pair(var(X), var(Y)), rev_pair(var(Y), var(X)))`

二、 等价公理： `equiv`

- **结构**: `equiv(TermA, TermB)`
- **作用**: 定义一条等价公理，声明两个项是等价的。这本身不是一个主动动作，而是影响 `rewrite` 规则匹配方式的背景条件。
- **参数**:
    - `TermA`, `TermB`: 两个被视为等价的、不含变量的“基项 (Ground Term)”。

- **机制**: 在尝试将 `rewrite` 规则的 `LHS` 与游戏中的项进行匹配时，如果 `TermA` 出现在匹配位置，引擎会认为 `TermB` 也在那里（反之亦然），从而发现更多可能的匹配。
- **示例**: `equiv(apple, red)`

3.3.2 结构/数据型构造函数 (Data Constructors)

- **说明**: 以下是在示例关卡中出现的、用于组织游戏状态的构造函数。它们本身没有特殊的内置逻辑，你可以自定义任何你喜欢的名称来构建关卡的数据结构。
- **示例分析**:
  - `puzzle_state(...)`: 作为包裹整个游戏状态（规则、公理、数据）的根节点，便于管理。
  - `data(...)`: 用于标记那些代表“状态”而非“规则”的项，纯粹为了关卡设计的可读性。
  - `pair(...)`, `rev_pair(...)`: 纯粹的数据结构，用于演示规则如何作用于数据。

3.3.3 特殊常量

- `placeholder`:
  - **类型**: 常量 (Constant Term)
  - **作用**: 在游戏界面上表现为一个可点击的“占位符”，允许玩家执行“构造”动作。
  - **机制**: 点击后会弹出一个模态框，让玩家输入一个新的项来替换它。新项的“代价”不能超过剩余的“预算”。

3.4 资源计算机制

- **分析来源**: 资源消耗逻辑由 `ApplyAction` 函数处理，而构造代价由 `Size(T)` 函数严格定义。
- **燃料 (Fuel)**: 每次成功执行一个 `rewrite` 动作，固定消耗 **1** 点燃料。
- **预算 (Budget) 与 项的代价 (Cost)**:
  - 构造新项时消耗的预算等于该项的“代价”。代价由 `Size(T)` 函数递归计算：
    1. **基础代价**: 任何项（常量、变量）的基础代价为 **1**。
    2. **复合代价**: 一个复合项的代价等于 **1** (为其自身的构造函数) **加上** 其所有子项的代价之和。
  - **计算示例**:
    - `cost(apple)` = 1
    - `cost(var(X))` = 1
    - `cost(pair(apple, green))`
      - = 1 (for `pair`) + cost(`apple`) + cost(`green`)
      - = 1 + 1 + 1 = **3**
    - `cost(rewrite(var(X), apple))`
      - = 1 (for `rewrite`) + cost(`var(X)`) + cost(`apple`)
      - = 1 + 1 + 1 = **3**

3.5 应用与注意事项

- **应用设置**: 在开发者面板中完成所有设置后，必须点击 **“应用设置”** 按钮来加载你的关卡。
- **格式化**: 在“当前项”和“目标项”的输入框中，你可以使用**换行和空格**来格式化你的代码，使其更易读。引擎在解析前会自动移除所有空白字符。
- **语法错误**: 无效的项语法会导致关卡加载失败。请仔细检查你的括号是否匹配、逗号是否正确使用。错误信息会显示在“应用设置”按钮旁边。