Architectural Specification: Computational Physics Validation for Automated Tool Extraction in Salvagnini P4 Systems

1. Executive Summary and System Architecture

The validation of automated manufacturing sequences for Computer Numerical Control (CNC) panel benders represents a critical intersection of computational geometry, rigid body kinematics, and material physics. Within the domain of the Salvagnini P4 system, the "Physics Gatekeeper" (Phase 4) serves as the definitive arbiter of feasibility. Its primary mandate is to ensure that the theoretical motions generated by the Phase 3 Sequencer can be executed in the physical world without catastrophic collision, tool entrapment, or violation of material constraints.

This architectural report delineates the algorithms and data structures required to simulate the "Trap Scenario"—a topological state where the Automatic Blankholder Adjustment (ABA) tool is geometrically enclosed within a formed box profile. Unlike traditional collision detection which checks static interference, this system must validate a dynamic extraction sequence involving tool contraction, Z-axis elevation, and Y-axis retraction under the influence of stochastic material behaviors such as springback and corner deformation. The proposed architecture utilizes a continuous collision detection (CCD) framework implemented in C++, leveraging the Separating Axis Theorem (SAT) for convex hull validation and dynamic bounding volume hierarchies (BVH) for real-time performance.

The complexity of the P4 system, with its manipulator kinematics, segmented ABA tooling, and interpolated bending blades, necessitates a simulation environment that transcends simple geometric bounding boxes.[1] We must model the machine not as a static entity but as a system of articulated rigid bodies with discrete state-spaces (e.g., the 5mm segmentation steps of the ABA).[3] Furthermore, the validation logic must account for the elastoplastic relaxation of the workpiece (springback), which alters the effective exit geometry, potentially turning a theoretically safe extraction into a collision event.[4]

2. Machine Topology and Kinematic Constraints

To engineer a valid physics engine, we must first rigorously define the kinematic chain and topological constraints of the Salvagnini P4. The simulation is only as accurate as the digital twin it operates upon.

2.1 The Automatic Blankholder Adjustment (ABA) System

The ABA is the primary variable in the extraction equation. Unlike static tooling, the ABA adapts its length to match the dimension of the panel being processed. However, this adaptation is not infinitely continuous; it is quantized, governed by the mechanical segmentation of the tool.

2.1.1 Discretization of Tool Width

The ABA comprises a central segment and variable side segments. The adjustment occurs in discrete steps, typically 5mm or similar increments depending on the specific P4 model configuration.[3] This discretization is critical for the validation algorithm. We cannot assume an arbitrary tool width W_{tool}; we must map the required width to the nearest available discrete step $W_{discrete}$ such that $W_{discrete} \leq W_{target}$.

The validation engine must query the machine state database to determine the *exact* configuration of the ABA.

$$W_{ABA}(t) = W_{center} + \sum_{i=1}^{n} S_{left,i} \cdot \delta_i + \sum_{j=1}^{m} S_{right,j} \cdot \delta_j$$

Where:

- W_{center} is the fixed width of the central segment.[3]
- $S_{k,i}$ is the width of the i-th segment.
- $\delta_i \in \{0, 1\}$ represents the engagement state of the segment.

If the calculated $W_{ABA}(t)$ exceeds the effective inner width of the box at any point during the extraction vector, a collision is flagged.

2.1.2 The Central Segment Constraint

A critical hard constraint is the physical dimension of the central segment. Even if all side segments are disengaged, the tool cannot shrink below W_{center}. Research indicates that for specific setups, the minimum panel width is constrained by this central unit and the manipulator's grip area.[6] If the box width W_{box} is less than the minimum collapsed width of the ABA W_{ABA_min}, extraction is physically impossible. This defines the lower bound of the "Feasibility Domain" for the Sequencer.

2.2 Manipulator and Blade Kinematics

The physics engine must also model the manipulator's grasp and the bending blades. The P4 manipulator rotates the sheet to present different edges to the bending line.[5]

- **Blade Interference:** The upper and lower blades have specific profiles allowing for positive and negative bends. During the "Trap" scenario, the blades typically retract to a "parked" position. However, the simulation must verify that the "Up" movement of the blankholder (Z-axis lift) does not drive the top of the formed box into the parked upper blade mechanism.[8]
- **Manipulator Collision:** When forming the final side, the manipulator holds the base of the box. The validation logic must check if the rotation of the manipulator causes the already formed flanges (specifically the return flanges) to collide with the ABA structure itself before the extraction sequence begins.

2.3 Topological Definition of the "Trap"

The "Trap" is not merely a geometric state; it is a topological event. In graph theory terms, the sheet metal part transitions from an open tree structure to a semi-closed cycle. When bending the final side of a box (Side 4), the ABA tool becomes topologically enclosed by Side 2 (front), Side 4 (back), and the flanges of Sides 1 and 3 (sides).

The extraction operation requires the ABA to contract laterally to clear the flanges of Sides 1 and 3. The simplified mathematical constraint is:

$$W_{ABA} < W_{Box} - 2 \times H_{Flange}$$

However, this simplification is dangerous for automation. It fails to account for:

1. **Return Flanges:** If Sides 1 and 3 have return flanges (C-bends), the effective exit width is significantly reduced. The tool must clear the *tips* of the return flanges, not just the web.
2. **Springback:** The flanges of Sides 1 and 3 may spring inward (toe-in), reducing the clearance.
3. **Corner Reliefs:** The geometry at the intersection of the flanges changes the collision hull, potentially creating snag points.

Therefore, the Phase 4 Validation Engine must implement a more rigorous check:

$$\text{IsSafe} = \forall P \in \text{ Path}_{extract}, \text{Collision}(\text{Tool}(t), \text{Part}(t, \text{Springback})) = \text{ False}$$

3. Computational Material Physics: Springback Integration

A geometric check based on CAD dimensions is insufficient for high-precision validation. The

"Physics Gatekeeper" must simulate the elastic recovery of the material to determine the *true* geometric constraints at the moment of extraction. Springback is the elastic recovery of the material upon unloading, leading to a change in the bend angle and radius.[4]

3.1 The Mechanics of Elastic Recovery

When the bending blades disengage, the stress state in the bent region relaxes. The metal, behaving as an elastoplastic material, releases the elastic component of the strain. This results in the flange moving from its loaded position (overbent) to its resting position.

The magnitude of this recovery is driven by the ratio of the bend radius (R) to the sheet thickness (T) and the material's yield strength (Y) relative to its Young's Modulus (E).[4]

$$\frac{R_i}{R_f} = 4\left(\frac{R_i Y}{ET}\right)^3 - 3\left(\frac{R_i Y}{ET}\right) + 1$$

Where R_i is the initial radius (under load) and R_f is the final radius (after release).

Implication for Extraction: Springback does not just change the angle; it shifts the spatial position of the flange tip. A change in angle $\Delta\theta$ results in a lateral displacement of the flange tip $\Delta x \approx L \cdot \sin(\Delta\theta)$, where L is the flange length. For deep boxes with long flanges, a small angular springback can result in a significant reduction of the exit width.

3.2 Calculating Effective Inner Width (W_{eff})

In a box closing scenario, we are concerned with the clearance between the *tips* of the previously formed flanges (Sides 1 and 3). We must derive the effective width W_{eff} available for the ABA tool.

Assume a nominal box width $W_{nominal}$. The flanges on Sides 1 and 3 have height H.
Due to springback, the flanges are not perfectly 90 degrees.

- **Case A (Spring-out / Obtuse):** If the bend relaxes to $> 90^\circ$, the flange tips move *outward*. This theoretically *increases* the clearance for the tool. However, the simulation must check if this outward movement causes the flange to collide with the blade holders during the bend of Side 4.
- **Case B (Spring-in / Toe-in):** This is the critical danger. If the material has residual stresses or if the "Return Flanges" point inward, the effective width decreases.

$$W_{eff} = W_{nominal} - 2 \times (T + R_{internal}) - 2 \times \delta_{inward}$$

Where δ_{inward} is the projection of the flange tip into the box volume.

The physics engine must calculate the position of the flange tip vector V_{tip} in the machine coordinate system.

$$V_{tip} = P_{bend\_origin} + \mathbf{R}_{springback} \cdot V_{flange\_length}$$

Where $\mathbf{R}_{springback}$ is the rotation matrix representing the relaxed state of the bend.

Algorithm Implication: The PartGeometry input to the validation step must be the *relaxed* mesh, not the theoretical CAD mesh. The validation engine applies a "Springback Modifier" to the mesh vertices before running collision checks.

3.3 Material-Specific Constants

The simulation must look up material properties to estimate the Springback Factor (K_s).

- **Mild Steel:** Low yield strength, minimal springback. $K_s \approx 0.99$.
- **Stainless Steel:** High yield strength, significant springback. K_s can range from 0.95 to 0.98 depending on rolling direction.[11]
- **Aluminum:** Lower modulus E, but generally behaves predictably.

The physics engine must define a Material class that encapsulates these properties and provides a GetSpringbackFactor(BendGeometry) method.

4. Geometric Topology: Corner Reliefs and Interferences

Corner reliefs are topological cuts made at the intersection of bends to prevent material tearing and allow for proper folding.[12] While primarily structural, they have a profound impact on validation logic because they modify the collision hull of the part.

4.1 Relief Types and Collision Hulls

The type of relief significantly alters the "Safe Zone" for extraction.

- **Square/Rectangular Relief:** Creates a clean rectangular void. This is the simplest for collision detection as it aligns well with AABB structures. It effectively increases the clearance at the corners.

- **Circular/Round Relief:** Creates a curved void. This requires mesh-based collision or approximated polygon testing.
- **Tear Relief:** This involves no material removal, just a slit. This is the most dangerous scenario for tool extraction. The material may deform unpredictably, creating burrs or overlapping flaps that protrude into the extraction path.[13]

Validation Logic:

If ReliefType == TEAR, the physics engine must apply a "Safety Penalty" to the clearance calculation.

$$\text{Clearance}_{req} = \text{Clearance}_{base} + \delta_{tear}$$

Where δ_{tear} accounts for the unpredictable protrusion of the torn edge. Conversely, for ReliefType == SQUARE, the engine can perform a boolean subtraction of the relief volume from the collision hull, potentially allowing the tool to pass closer to the corner than would otherwise be calculated.

4.2 The "Hook" Effect

A critical failure mode during extraction is the "Hook" effect. As the tool retracts (Y-axis) while lifting (Z-axis), a corner of an ABA segment might catch on the edge of the relief cut.

- **Vector Analysis:** The extraction vector E must be strictly contained within the "Safe Cone" defined by the corner geometry.
- **Re-entrant Geometry:** If the relief is a keyhole shape (circular with a slot), the tool could theoretically enter the keyhole and become mechanically interlocked. The validation algorithm must check for re-entrant geometry along the extraction path to ensure smooth egress.

5. Algorithmic Core: Collision Detection Framework

To implement the validation logic in C++, we require a high-performance collision detection framework. Given the real-time constraints of industrial simulation (validating thousands of branches in a search tree), efficiency is paramount.

5.1 Data Structures: AABB vs. OBB

- **AABB (Axis-Aligned Bounding Box):** Fast to compute and aligned with global axes (X, Y, Z). These are excellent for broad-phase culling—quickly determining if the tool is even near a flange.

- **OBB (Oriented Bounding Box):** These boxes rotate with the object. They are necessary for representing the sheet metal part (which rotates) and the flanges (which are angled due to springback).
- **Mesh (Convex Hull):** The most accurate representation, required for the ABA tool which has a complex stepped profile.

Recommendation: Use a **Two-Phase Detection System**.

1. **Broad Phase:** Dynamic AABB Tree (BVH) to efficiently eliminate non-interacting faces.
2. **Narrow Phase:** SAT (Separating Axis Theorem) between the OBB of the tool segments and the OBB of the flange segments.[14]

5.2 The Separating Axis Theorem (SAT) Implementation

For the narrow phase, SAT is ideal for convex shapes like box flanges and tool segments.
Theorem: Two convex shapes do not intersect if and only if there exists a separating axis on which their projections do not overlap.[14]

Axes to Test (3D OBBs):

1. Face normals of Body A (3 axes).
2. Face normals of Body B (3 axes).
3. Cross products of edges of A and edges of B ($3 \times 3 = 9$ axes).
   Total 15 axes to test.

C++ Implementation Logic:

C++

```cpp
// C++ Pseudocode for SAT Check
bool CheckCollision(OBB& tool, OBB& flange) {
    // 1. Compute rotation matrix expressing flange in tool's coordinate frame
    // 2. Compute translation vector
    // 3. Test 15 axes:
    //    - 3 axes of tool
    //    - 3 axes of flange
    //    - 9 cross product axes
    // If any overlap test returns false (gap found), return false immediately (No Collision).

    // Example: Test Axis Ax
    float ra = tool.e; // radius of A
    float rb = flange.e*AbsR + flange.e*AbsR + flange.e*AbsR;
    float t = abs(T); // projection of translation
    if (t > ra + rb) return false; // Separating axis found

    //... repeat for all 15 axes...

    return true; // Collision confirmed
}
```

5.3 Swept Volume for Extraction Simulation

Static collision detection only checks a specific snapshot in time. However, extraction is a continuous motion. We must validate the *path* to prevent "tunneling," where a fast-moving tool might jump through a thin flange without detection in discrete time steps.

We employ **Swept Volume** or **Continuous Collision Detection (CCD).**[15] The extraction vector $V_{extract}$ defines the sweep direction. We construct a new volume that represents the union of the tool geometry at t_{start} and t_{end}.

- For an AABB, the swept volume is a larger AABB encompassing the start and end positions.
- For precise validation, we can perform a "convex cast" of the tool shape along $V_{extract}$ against the static geometry of the box.

6. Tool Extraction Validation Algorithm (The "Gatekeeper")

This section details the specific algorithm to be implemented in the Phase 4 engine.

6.1 Input Parameters

- **PartState**: Mesh data of the formed part, vertices V, faces F.
- **ToolState**:
  - CurrentWidth: W_{AB}.
  - SegmentConfig: List of active segments (e.g.,).
  - Position: (x, y, z) coordinates.
- **ExtractionPath**: Sequence of vectors (e.g., $v_1 =$, $v_2 =$).

6.2 The Validation Pipeline

Step 1: Springback Mesh Deformation

Before checking collision, deform the PartState mesh to account for springback.

For each flange f:

1. Calculate K_s based on material props (E, Y).
2. Rotate vertices of f around the bend axis by $\Delta\theta = \theta_{target} \times (1 - K_s)$.
3. Update the collision volume (OBB) of the flange.

Step 2: Effective Exit Width Check (The Heuristic Filter)

Perform a fast scalar check before expensive collision detection.

$$W_{exit} = \text{GetMinInnerWidth}(PartState)$$

$$W_{tool\_collapsed} = \text{GetABACollapsedWidth}(ToolState)$$

If $W_{tool\_collapsed} > W_{exit} - \text{Tolerance}$:

Return FAILURE_TRAP_DETECTED (Geometric impossibility).

Step 3: Discrete Tool Sizing

The ABA cannot be an arbitrary width. It must snap to available segments.

The Sequencer might request $W = 500mm$.

The Physics Engine checks: Can we form 500mm?

Available segments: Central(200), Side(5, 10, 20...).

Constraint: Is there a combination $\sum S_i \approx 500$ that fits?

If the required contracted width for extraction is 400mm, can the ABA actually contract to 400mm from its current state?

- *Note:* The ABA adjusts *in-cycle*.[1] We must verify the time/distance required to contract.

Step 4: Swept Extraction Simulation

Simulate the retraction path.

C++

```cpp
bool ValidateExtraction(PartMesh& part, ToolMesh& tool, vector<vec3> path) {
  ToolMesh currentTool = tool;
  for (vec3 step : path) {
    // Create Swept Volume of tool along vector 'step'
    Volume sweptTool = CreateSweptVolume(currentTool, step);

    // Check intersection with Part
    if (CheckIntersection(sweptTool, part)) {
      return false; // Collision detected along path
    }

    // Move tool to next waypoint
    currentTool.Translate(step);
  }
  return true;
}
```

6.3 Handling Return Flanges (The "C" Profile)

When bending a "C" profile or a box with return flanges, the exit is constricted.

The validation logic must identify "Overhanging Geometry."

- Raycast vertically from the tool edges up. If the ray hits a flange (the return flange), the vertical lift (Z-axis) is blocked.
- **Correction Logic:** If Z-lift is blocked, the engine must propose a **Lateral Shift (X-axis)** or **Y-retract** *before* the lift.
- The Sequencer proposes: Bend -> Lift -> Retract.
- Physics Engine rejects.
- Physics Engine Counter-proposal: Bend -> Contract ABA -> Lateral Shift (center tool) -> Lift -> Retract.

7. C++ Implementation Strategy

The implementation relies on rigorous object-oriented design to manage the complexity of the machine state.

7.1 Class Structure

```C++
class GeometricEntity {
public:
    virtual AABB GetAABB() const = 0;
    virtual OBB GetOBB() const = 0;
    //...
};

class ABATool : public GeometricEntity {
private:
    float currentWidth;
    std::vector<Segment> activeSegments;
    float centralSegmentWidth; // Constraint from
public:
    bool Resize(float newWidth);
    Volume GetSweptVolume(Vector3 direction);
};

class FormedPanel : public GeometricEntity {
private:
    Mesh deformedMesh;
    MaterialProperties matProps;
public:
    void ApplySpringback(); // Updates deformedMesh
    bool CheckClearance(const ABATool& tool);
};

class ValidationEngine {
public:
    ValidationResult ValidateStep(const Action& step, const MachineState& state);
};
```

7.2 Algorithms for Safety Verification

Algorithm 1: Safe Width Calculation with Springback

1. Retrieve W_{box} (Inner width from CAD).
2. Retrieve H_{flang} (Flange height).
3. Calculate θ_{sb} (Springback angle deviation).

   - $\theta_{sb} = f(Material, R, T)$.
4. Calculate Intrusion I due to springback:
   - $I = H_{flange} \times \sin(\theta_{sb})$.
5. Calculate $W_{safe} = W_{box} - 2 \times I - \text{SafetyMargin}$.
6. If $ABA_{min\_width} > W_{safe}$, return FAIL.

Algorithm 2: Vector Logic for Retraction

The extraction usually follows a vector $V = (0, -Y, Z)$.

1. Project the Tool Hull onto the plane perpendicular to V.
2. Project the Flange Hulls onto the same plane.
3. Check 2D overlap.
4. If overlap exists, check depth along V.

7.3 Accounting for Corner Reliefs in Math

When calculating W_{safe}, the corner relief acts as a modifier.

If Relief == Square, the effective H_{flang} at the corner is 0 for the height of the relief.

The algorithm can define a SafeZone at the corners where the tool *can* technically intersect the theoretical flange plane because the material is removed.

- **Boolean Logic:** Collision = Intersect(Tool, Flange) AND NOT Intersect(Tool, ReliefVolume).
- This allows the simulation to approve "tight" extractions where the tool corners pass through the relief voids.

8. Error Handling and Correction Injection

When the ValidateExtraction returns false, the Physics Gatekeeper must inform the Sequencer *why* and propose *alternatives*.

8.1 Failure Modes

1. **trap_width:** The box is physically too narrow for the ABA even at min width.
   - *Action:* Discard branch. Mark part as "Unfeasible on P4" (Requires manual brake or P2 with narrower tool if applicable).
2. **trap_height:** The return flange blocks vertical lift.
   - *Action:* Inject Action::ABA_Resize. Contract tool further if possible.

    - *Action:* Inject Action::Manipulator_Shift. Move panel to center tool in the gap.
3. **collision_springback:** Theoretical clearance exists, but springback closes the gap.
    - *Action:* Update Bend Sequence to use "Negative Bend" pre-compensation (overbending) to ensure final clearance? No, overbending usually *worsens* the clearance during the extraction of the tool (tool is inside the acute angle).
    - *Correction:* Use a "Sacrificial" extraction path? No.
    - *Correction:* Recommend using **CLA Tools** (Compo Length Adjustment) which are modular auxiliary blades that can engage/disengage.[8] If the active tool width is reduced by dropping segments, extraction becomes possible.

8.2 The "Repo" Strategy (Repositioning)

If the manipulator cannot rotate the part due to collision with the tool during the "Trap" exit:

1. **Release:** Blankholder releases.
2. **Regrip:** Manipulator releases, moves to a new grasp point (if accessible), and regrips.
3. **Retract:** Attempt extraction with new pivot point.
   The Physics Engine must validate the stability of the part during the "Release" phase (gravity check).

9. Simulation Verification and Data-Driven Refinement

The final stage of the architecture is the simulation loop.

9.1 Discrete Time Simulation

We simulate the extraction in time steps Δt.

$$P_{tool}(t + \Delta t) = P_{tool}(t) + V_{extract} \times \Delta t$$

At each step:

1. Update Tool OBB.
2. Run SAT Check against Part OBB Tree.
3. If Collision: Backtrack to t and try to resolve (e.g., nudge Y/Z).

9.2 Optimization

To optimize the $O(N^2)$ collision checks:

- **Spatial Hashing:** Map the part flanges to a voxel grid. Only check tool segments against occupied voxels.
- **Bitmasking:** Use bitmasks for ABA segments. ToolMask & CollisionMask.

10. Conclusion

The "Trap Scenario" in box closing on a Salvagnini P4 is a geometric certainty that must be managed, not avoided. By constructing a Physics Engine that incorporates:

1. **Elastoplastic Modeling:** Springback-adjusted effective widths.
2. **Discrete Kinematics:** 5mm stepped ABA sizing.
3. **Advanced Collision Detection:** Swept OBB using SAT.
4. **Topological Awareness:** Corner relief boolean logic.

We create a validation layer that ensures 100% physically viable NC code. This architecture moves the P4 from a machine that executes commands to a system that understands the physics of its own operation, acting as the ultimate gatekeeper for automated production.

Reference Table: Mathematical Symbols & Definitions

Symbol	Definition	Source Relevance
W_{AB}	Current Width of Automatic Blankholder	7
W_{bo}	Inner Width of the formed box	17
H_{flang}	Height of the return flange	19
K_s	Springback Factor	4
ϵ	Safety Clearance (Skin Width)	20
E	Extraction Vector (Lift + Retract)	21
ABA_{mi}	Minimum physical width of ABA (central segment)	3

Code Logic: Tool Extraction Validation (C++ Prototype)

C++

```cpp
/**
 * Validates if the ABA tool can be extracted from a closed box profile.
 *
 * @param boxWidth Inner width of the box (mm).
 * @param flangeHeight Height of the return flanges (mm).
 * @param toolWidth Current width of the ABA tool (mm).
 * @param springbackAngle Estimated springback deviation (degrees).
 * @param cornerReliefType Type of corner relief (SQUARE, ROUND, TEAR).
 * @return True if extraction is physically feasible.
 */
bool IsExtractionSafe(double boxWidth, double flangeHeight, double toolWidth,
                      double springbackAngle, ReliefType cornerReliefType) {

    // 1. Calculate Geometric Constraints
    double clearance = 2.0; // Standard machine clearance (mm)

    // 2. Adjust for Springback
    // If flanges spring IN (acute angle), effective width decreases.
    // If flanges spring OUT (obtuse), effective width increases (usually safer).
    // Worst case: Return flanges typically toe-in or interfere vertically.
    double springbackIntrusion = flangeHeight * sin(DegToRad(springbackAngle));

    // 3. Adjust for Corner Reliefs
    // A tear relief effectively reduces safe width due to burr unpredictability.
    double reliefPenalty = 0.0;
    if (cornerReliefType == TEAR) {
        reliefPenalty = 1.5; // mm safety buffer
    }

    // 4. Calculate Effective Exit Width
    double effectiveWidth = boxWidth - (2 * flangeHeight) - (2 * springbackIntrusion) -
reliefPenalty;

    // 5. Compare against Tool Width
    if (toolWidth + clearance < effectiveWidth) {
        return true; // Safe
    } else {
        // Try to Resize ABA
        double minABA = GetMinABASegmentWidth(); // e.g., 105mm or central segment width
        if (minABA + clearance < effectiveWidth) {
            // Suggest Resize Action
            Log("Correction Needed: Resize ABA to minimum.");
            return true; // Feasible with correction
        }
    }

    return false; // Trap detected
}
```

11. Detailed System Kinematics and Dynamic Constraints

The validation of the "Trap Scenario" requires a profound understanding of the Salvagnini P4's internal kinematics. The interaction between the Automatic Blankholder Adjustment (ABA) and the panel is not merely a static geometric fit; it is a dynamic event governed by the machine's axis velocities, accelerations, and the discrete nature of its tooling.

11.1 The Anatomy of the ABA System

The ABA tool is the central protagonist in the extraction drama. It is not a monolithic block but a highly sophisticated, segmented mechanism designed to expand and contract. Understanding its anatomy is crucial for accurate simulation.

11.1.1 Discrete Segmentation Logic

The ABA is composed of a fixed central segment and a series of movable side segments. The central segment typically has a width of 200mm or similar, depending on the machine model (e.g., P4-2525 vs P4-3125).[3] Flanking this center are modular segments of varying widths—often 5mm, 10mm, 20mm, etc.—that can be engaged or disengaged.

The validation algorithm must model this discreteness. We cannot command the ABA to be 423mm wide. It must snap to a valid combination of segments, say 425mm or 420mm.

- **Safety Implication:** If the required safety clearance suggests a max tool width of 423mm, and the machine can only achieve 420mm or 425mm, the system must force the tool to 420mm. If the tool logic defaults to the "nearest" (425mm), a collision will occur. The algorithm must always floor the width to the nearest valid segment combination that is *less than* the safe limit.

$$W_{command} = \max\{W \in \mathbb{S}_{segments} \mid W \leq (W_{box} - \text{Clearance})\}$$

Where $\mathbb{S}_{segments}$ is the set of all achievable ABA widths.

11.1.2 Actuation Dynamics

The ABA segments are actuated pneumatically or electrically. This actuation takes time—typically milliseconds to seconds depending on the magnitude of the change. In a "Trap" scenario, the tool resize happens *inside* the closed box.

- **Time-Step Simulation:** The physics engine must verify that the ABA has fully contracted to the safe width *before* the Z-axis lift begins. A race condition where the lift starts while the segments are still retracting would cause a collision with the return flanges.
- **Sequence Logic:**
  1. Command: ABA_Resize(TargetWidth)
  2. Wait: WaitForSignal(ABA_In_Position)
  3. Command: Z_Axis_Lift()
     The validation engine treats this as a dependency graph node. Action::Bend is a predecessor to Action::ABA_Resize, which is a predecessor to Action::Extract.

11.2 Manipulator Kinematics (DPM vs. Clamp)

The extraction sequence is heavily influenced by how the part is held. The P4 typically uses a manipulator with clamping jaws or the DPM (Digital Part Manipulator) with suction cups.[6]

11.2.1 The Grip Constraint

When the ABA contracts, the panel must be held stationary by the manipulator.

- **Clamp Interference:** If standard clamps are used, they grip the edge of the sheet. In a box closing scenario, the "back" of the box (Side 3) is already bent. The manipulator must grip the *flange* of Side 3 or the base of the box.
- **Collision Hull:** The manipulator itself has a collision hull. When the ABA retracts (Y-axis), it moves towards the manipulator. The validation engine must check for collisions between the *back* of the ABA tool and the *face* of the manipulator.
- **DPM Advantage:** The suction cup manipulator (DPM) allows for gripping the flat surface of the panel, avoiding edge interference. This often allows for deeper boxes or tighter extraction paths. The physics engine must switch its "Manipulator Model" based on the machine configuration provided in the MachineState object.

11.2.2 Rotational Constraints

To extract the tool, the part might need to be rotated or shifted laterally to align the ABA with the widest part of the exit (e.g., diagonally). The P4 manipulator can rotate the sheet.

- **The "Diagonal" Exit:** For square boxes, the width is constant. But for rectangular boxes, or boxes with complex flange geometries, rotating the part slightly might open a wider path for the tool.

- **Algorithm:** The IsExtractionSafe function should not just check the alignment at $\theta = 0$. It should explore a small rotational perturbation $\theta \in [-\alpha, +\alpha]$ to see if a collision-free path exists. This is akin to the "Piano Mover's Problem" in robotics path planning.

12. Geometric Modeling of the Sheet Metal Part

The input PartGeometry is not a simple mesh; it is a topological graph representing a folded sheet metal structure.

12.1 Graph-Based Topology

We model the part as a graph $G(V, E)$ where:

- **Vertices (V):** Planar faces of the sheet metal (The base, the sides, the flanges).
- **Edges (E):** The bend lines connecting the faces.

A "Closed Profile" or "Box" is detected by analyzing the adjacency matrix of this graph. If the graph contains a cycle of faces that enclose a volume (topologically similar to a cylinder or open box), the "Trap" logic is triggered.

- **Face Normals:** We compute the normal vectors n_i for all faces. If a set of faces has normals pointing towards a common centroid, they define an enclosure.

12.2 B-Rep vs. Mesh Representation

For collision detection, we need a dual representation:

1. **B-Rep (Boundary Representation):** Exact mathematical definition (planes, cylinders for bends). Used for precise clearance calculation.
2. **Tessellated Mesh:** Triangulated approximation used for the fast broad-phase collision detection (AABB/OBB).

The simulation engine converts the B-Rep to a Mesh at the start of Phase 4, applying the "Bend Angles" from the Phase 3 sequencer. Crucially, this mesh is *dynamic*. As the bend angle changes (during the simulation of the bend itself) or relaxes (springback), the mesh vertices are updated via a vertex shader-like compute kernel.

12.3 Corner Topology and boolean Operations

The corners of the box are complex. They are where three faces meet (Base, Side 1, Side 2). The "Corner Relief" is a boolean subtraction from this junction.

- **Algorithm:**
    1. Generate the "Ideal" corner geometry (sharp intersection).

2. Generate the "Relief" volume (Cylinder for round, Box for square) centered at the bend intersection.
3. Perform Boolean Difference: Corner_Geo = Ideal_Geo - Relief_Volume.
4. The resulting Corner_Geo is the actual collision hull.

This boolean operation allows the simulation to accurately predict if the sharp corner of the ABA tool will "poke through" a square relief (safe) or collide with the edge of a tear relief (unsafe).

13. Advanced Material Physics: Microstructure and Springback

The behavior of the metal during extraction is dictated by its elastoplastic properties. To truly validate the process, we must understand *why* springback occurs and how to model it.

13.1 Elastoplastic Deformation Mechanics

When the blade bends the sheet, the material fibers on the outside of the neutral axis are in tension, and those on the inside are in compression.

- **Plastic Zone:** The outer fibers exceed the Yield Strength (Y) and deform plastically (permanently).
- **Elastic Core:** The material near the neutral axis may remain in the elastic region (below Yield Strength).
- **Unloading:** When the tool retracts, the elastic stresses try to resolve. The "Elastic Core" tries to straighten the sheet, while the "Plastic Shell" resists. The equilibrium point is the sprung-back angle.

13.2 Modeling Variable Springback

A static K_s factor is often insufficient because springback varies with:

1. **Rolling Direction:** Sheet metal is anisotropic. Bending "with the grain" vs. "against the grain" results in different springback magnitudes.[11] The validation engine should ideally know the sheet orientation.
2. **Work Hardening:** If the material has been pre-worked or if the bend radius is very tight (coining), the yield strength locally increases, changing the springback.
3. **Variable Thickness:** Standard tolerances in sheet thickness ($T \pm 0.05mm$) can cause significant variations in angle.

Robust Validation: The engine should run a **Monte Carlo Simulation**.

- Run the extraction validation 100 times.

- Vary T (Thickness) and Y (Yield Strength) within statistical tolerance limits (Gaussian distribution).
- If *any* of the 100 simulations result in a collision, flag the sequence as "High Risk" or apply a larger safety margin.

13.3 The "Negative Springback" Anomaly

In certain geometries, particularly with "Coining" (bottoming out the punch) or specific return flange configurations, the flange may spring *in* (towards the bend). This is often caused by the displacement of the neutral axis or complex residual stresses.[4]

- **Danger:** Negative springback reduces the exit width.
- **Policy:** The Physics Engine acts conservatively. It assumes the *worst-case* springback direction for the given geometry. For a box, the worst case is always "Toe-In" (flanges closing the gap). The validation equation uses the "Toe-In" vector for clearance checks unless empirical data confirms otherwise.

14. Simulation of the Extraction Path (Vector Logic)

The extraction is a vector field problem. We need to find a path $P(t)$ for the tool center C_{tool} such that the tool volume V_{tool} never intersects the part volume V_{part}.

14.1 The Canonical Extraction Vectors

For a Salvagnini P4, the extraction is typically a superposition of two primary vectors:

1. $Z_{lift} = [0, 0, +1]$ (Vertical Lift)
2. $Y_{retract} = [0, -1, 0]$ (Horizontal Retract away from the bend)

The combined vector is often a sequence, not a simultaneous move, to avoid "hooking."

Sequence:

1. **Contract:** $\Delta Width_{ABA} < 0$. (Resize tool).
2. **Back-off:** $\Delta Y \approx 1 - 2mm$. (Move tool slightly away from the back flange to release pressure).
3. **Lift:** $\Delta Z > H_{flange}$. (Lift tool above the return flanges).
4. **Retract:** $\Delta Y \rightarrow \text{Home}$. (Move tool fully out).

14.2 Computing the "Escape Cone"

In complex geometries (e.g., non-90 degree boxes, trapezoidal shapes), the standard Z-then-Y path might fail. The physics engine computes the **Minkowski Difference** between the Tool and the Part.

$$M = P \ominus T = \{p - t | p \in P, t \in T\}$$

The origin O represents a collision. To escape, we need a vector v such that the ray from O along v exits M without intersecting geometry.

- In 3D, this defines an "Escape Cone" of valid vectors.
- If the machine's kinematic axes (Z and Y) do not lie within this cone, extraction is impossible without rotating the part (Manipulator Rotation).

14.3 Swept Volume Implementation Details

To prevent tunneling, we use the "Convex Cast" method.

Given Tool Convex Hull H_{tool} and displacement vector d.

We want to find the first time of impact $t \in $.

We support vertices of H_{tool} along d.

This effectively creates a prism with caps defined by H_{tool} at $t = 0$ and $t = 1$.

We test this prism against the static BVH of the Part.

This ensures that even thin flanges or small burrs from tear reliefs are detected if the tool path sweeps through them.

15. The "Sequencer" Interaction and Feedback Loop

The Physics Gatekeeper is not a passive verify-only step. It is an active component of the planning loop (Phase 3 -> Phase 4 loop).

15.1 Feedback Mechanisms

When ValidateExtraction() returns false, it provides an ErrorObject:

- ErrorType: COLLISION, TRAP_DIMENSION, KINEMATIC_LIMIT.
- Location: (x, y, z) of the impact.
- InterferenceDepth: Magnitude of the overlap.

- SuggestedCorrection: (e.g., "Decrease Tool Width", "Rotate Part 90 deg").

15.2 Search Tree Integration (A*)

The Sequencer uses an A* or RRT (Rapidly-exploring Random Tree) algorithm to find the bend sequence. The Physics Engine acts as the cost function or edge validator.

- If IsSafe == True: Cost = Cycle Time.
- If IsSafe == False: Cost = ∞ (Edge pruned).

However, calculating full physics for every branch is expensive.

Optimization: "Lazy Evaluation."

1. Sequencer builds the tree using simplified bounding boxes.
2. Once a candidate full path is found, the Physics Engine runs the rigorous simulation.
3. If it fails, the specific edge is invalidated, and the Sequencer replans.

16. Detailed C++ Architecture and Optimization

The implementation of this system requires a robust C++ architecture designed for performance and extensibility.

16.1 Memory Management and Data Locality

- **Data-Oriented Design (DOD):** Instead of heavy OOP hierarchies for every mesh vertex, use flat arrays (std::vector<vec3> vertices, std::vector<int> indices). This maximizes CPU cache coherence during collision detection loops.
- **SIMD Intrinsics:** Use SSE/AVX instructions for the SAT axis tests. Testing 15 axes involves many dot products. SIMD can perform 4 or 8 dot products in parallel, significantly speeding up the narrow-phase check.

16.2 Threading Model

- **Parallel Validation:** If the Sequencer generates 50 candidate sequences, the Physics Engine can validate them in parallel using a thread pool (e.g., std::async or OpenMP).
- **Job System:** Break the validation into jobs: "Update Mesh Job", "Broad Phase Job", "Narrow Phase Job". This allows the engine to scale across all available CPU cores.

16.3 Logging and Visual Debugging

- **Telemetry:** The engine must log the "Safety Margin" of every successful extraction. If a sequence succeeds with 0.1mm clearance, it should be flagged as "Marginal" for operator review.
- **Visualizer Interface:** The engine should implement a socket interface to stream the collision geometry (OBBs, Swept Volumes) to a 3D visualizer (e.g., OpenGL/Vulkan) so

engineers can "see" why a specific sequence failed.

17. Conclusion and Future Outlook

The validation of tool extraction in the Salvagnini P4 represents the frontier of "Physics-Based Manufacturing." By moving beyond simple geometric checks and integrating elastoplastic material modeling, discrete kinematic constraints, and advanced continuous collision detection, we create a system that guarantees process safety in a "Batch One" environment.

This architecture transforms the P4 from a rigid execution machine into an adaptive, intelligent system capable of understanding its own physical limitations. The "Trap Scenario," once a source of manual programming headaches and machine crashes, becomes a solvable constraint satisfaction problem, handled automatically by the rigorous logic of the Physics Gatekeeper. This ensures not just the physical integrity of the machine and the part, but the economic viability of the entire automated production line.

18. References

- [6] Salvagnini P4 Product Specification, Manipulator Options.
- Universal Bending Tools and ABA Kinematics.
- [8] P4 User Manual: CLA and Auxiliary Tools.
- [8] Automatic Blankholder Adjustment (ABA) Technical Data.
- [3] P4 Technical Specifications: Segment Increments.
- P4 Bending Cycle Description.
- [4] Understanding Springback in Sheet Metal.
- [9] Bending Springback Calculation Factors.
- [12] Protolabs Design Guide: Bend Reliefs.
- [13] SendCutSend Guide to Bend Relief Geometry.
- [7] P2/P4 Minimum Part Width Constraints.
- [5] P2/P4 ABA Segmentation Details.
- [4] Springback Factor (K_s) Derivation.
- [14] Separating Axis Theorem (SAT) Implementation Details.
- [3] Salvagnini P4 ABA Central Segment Dimensions.
- [20] Sheet Metal Design Guidelines: Clearances.
- [25] 247TailorSteel Bending Guidelines: Collision Avoidance.
- [10] MIT OpenCourseWare: Springback Mechanics.
- [17] Salvagnini P4 Video Analysis: Box Bending.
- [21] Italian Machines: Panel Bender Cycle Description.
- [7] P2/P4 Return Flange Extraction Limits.
- [13] Tear Relief vs. Round Relief: Manufacturing Implications.
- [22] Cleaning and Maintenance Manual: ABA Segment Assembly.

- [15] Swept AABB Collision Detection Algorithms.
- [18] Dalsin Industries: Salvagnini P4 Capabilities.
- [19] Approved Sheet Metal: Flange Height Formulas.
- [14] SAT vs. GJK for Convex Collision.
- [23] Grain Direction and Anisotropy in Bending.
- [11] Protocase: Material Specific Springback Data.
- [16] Continuous Collision Detection Implementation.

Works cited

1. Salvagnini panel bending: P4-2520 bends 4 different parts - YouTube, accessed February 3, 2026, https://www.youtube.com/watch?v=gOMHcSt2yQ0
2. Universal Sheet Metal Panel Bending Machine - Salvagnini, accessed February 3, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders
3. The automatic hybrid adaptive Panel Bender. - Sanson Machinery Group, accessed February 3, 2026, http://www.sansonmachinery.com/wp-content/uploads/2013/12/P4Xe-UK-3962020312.pdf
4. Understanding Springback In Sheet Metal Bending - SC Machinery - SC (Shenchong), accessed February 3, 2026, https://www.shen-chong.com/understanding-springback-in-sheet-metal-bending/
5. The flexible bending solution. - Sp-Tech, s.r.o., accessed February 3, 2026, https://www.sp-tech.cz/static/soubory/stranka-71/prospekt-salvagnini-p2-152.pdf
6. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 3, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
7. Lean & Compact Panel Bender - Salvagnini P2, accessed February 3, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P2
8. p4-benders.pdf - Empire Machinery, accessed February 3, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf
9. Bending Springback Calculator - CustomPartNet, accessed February 3, 2026, https://www.custompartnet.com/widgets/bending-springback
10. Solution Homework #5: Springback 2.008 Design and Manufacturing II Spring 2004 Out: March 31 Due: April 07 Problem 1 As demonst, accessed February 3, 2026, https://ocw.mit.edu/courses/2-008-design-and-manufacturing-ii-spring-2004/4c25fa18116b92aa429b85a4a188e92f_hmewrk_05_sol.pdf
11. Understanding Sheet Metal Springback - Protocase, accessed February 3, 2026, https://www.protocase.com/blog/2025/10/10/understanding-sheet-metal-springback/
12. How and When to Add Bend Reliefs to Sheet Metal Parts - Protolabs, accessed February 3, 2026, https://www.protolabs.com/resources/blog/bend-relief/
13. How to Design Bend Reliefs & Corner Reliefs for Sheet Metal Parts -

SendCutSend, accessed February 3, 2026, https://sendcutsend.com/blog/guide-to-designing-bend-reliefs/
14. The Math Behind Bounding Box Collision Detection - AABB vs OBB(Separate Axis Theorem), accessed February 3, 2026, https://dev.to/pratyush_mohanty_6b8f2749/the-math-behind-bounding-box-collision-detection-aabb-vs-obbseparate-axis-theorem-1gdn
15. Using Swept AABB to detect and process collision - Amanotes, accessed February 3, 2026, https://www.amanotes.com/post/using-swept-aabb-to-detect-and-process-collision
16. Swept AABB Collision Detection Using the Minkowski Difference - Kenton Hamaluik, accessed February 3, 2026, https://blog.hamaluik.ca/posts/swept-aabb-collision-using-minkowski-difference/
17. Salvagnini panel bender: smallest box on P2-1620 - YouTube, accessed February 3, 2026, https://www.youtube.com/watch?v=8BtzeuPImB8
18. Salvagnini P4 Sheet Metal Panel Bending Design Consideration - Dalsin Industries, accessed February 3, 2026, https://www.dalsinind.com/salvagnini-tech-specs.html
19. Sheet Metal Flange Height Formula for Proper Forming (2026), accessed February 3, 2026, https://www.approvedsheetmetal.com/blog/use-this-flange-formula-for-sheet-metal-forming
20. Sheet Metal Bending Design Tips | Xometry Pro, accessed February 3, 2026, https://xometry.pro/en/articles/sheet-metal-bending-design-tips/
21. The Panel Bender: Your Ultimate Guide - Italian Machinery Association, accessed February 3, 2026, https://www.italianmachines.eu/en/publications/437/the-panel-bender-your-ultimate-guide
22. 14cleaning The Blankholder Segments | PDF | Screw | Industrial Processes - Scribd, accessed February 3, 2026, https://www.scribd.com/document/422736346/14Cleaning-the-Blankholder-Segments
23. Grain direction's effect on sheet metal bending - The Fabricator, accessed February 3, 2026, https://www.thefabricator.com/thefabricator/article/bending/grain-directions-effect-on-sheet-metal-bending
24. A Review on Springback Effect in Sheet metal Forming Process - ASDF EDLIB, accessed February 3, 2026, https://edlib.net/2015/icssccet/ICSSCCET2015011.pdf
25. Guidelines for bending | 247TailorSteel, accessed February 3, 2026, https://247tailorsteel.com/en/submission-guidelines/guidelines-for-bending