KI-gestütztes CAD-Design: Entwicklung und Software-Integration einer CAD-Grammatik zur Erforschung von CAD-LLMs

AI-based CAD-Design: Development and Integration of a CAD-grammar for the development of CAD-LLMs

Leon Josef Stahr[1,*], Alexander Hasse[1]

[1] Chemnitz University of Technology

** Korrespondierender Autor:*
Leon Josef Stahr
Reichenhainer Straße 70
09126 Chemnitz
☎ *0351/531 39258*
✉ *leon-josef.stahr@mb.tu-chemnitz.de*

Abstract

The usage of feature- and text-based large language models (LLMs) in computer assisted design (CAD) promises increased productivity and effectiveness of designers. Due to the lack of adequate training data, CAD-LLMs generate syntactically invalid CAD-text. We propose a CAD rule set (CAD-grammar) that, when checked, ensures the syntactic validity of the generated text.
We demonstrate the usage of the CAD-grammar, by integrating it into an interface (CAD-Tool), that connects the User of a CAD-Software with a CAD-LLM.
The CAD-Tool enables direct control of the CAD-LLM and free translation between CAD-text and CAD-model. Furthermore, the CAD-Tool checks the CAD-Grammar of any CAD-LLM-Output before translating it into a CAD-Model. This catches and documents issues and ensures valid CAD-Texts.

Keywords

Generative CAD, LLM, rule-based design, CAD grammar

 | DOI: 10.35199/dfx2026.14

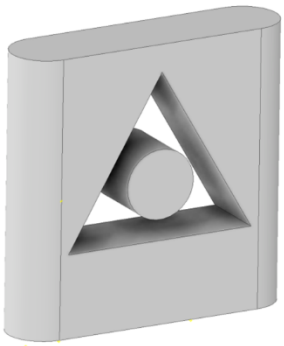
1. Einleitung

Die Anwendung von Large-Language-Modellen (LLMs) verspricht auch für Anwendungen des Computer Assisted Designs (CAD) viele Vorteile, u.a. im Bereich der Automatisierung und Flächenrückführung. Aufgrund ihrer Leistungsfähigkeit werden dafür häufig allgemeine vortrainierte LLMs wie Llama, Claude o.ä. verwendet. Aufgrund ihrer geringen Spezialisierung ist ihrer Verwendung häufig erst dann sinnvoll, wenn sie mittels „Fine-Tuning“ für eine spezielle Aufgabe trainiert werden oder durch „retrieval augmented generation“ (RAG) auf domänenspezifisches Wissen zurückgreifen können. Ein für CAD-Anwendungen trainiertes und spezialisiertes LLM wird im Folgenden *CAD-LLM* genannt. Damit derartige LLMs ihr Potential in CAD-Anwendungen entfalten können, müssen sie mit domänenspezifischen (CAD-)Daten trainiert werden.

Für die Anwendungen der Flächenrückführung und Automatisierung ist dabei besonders die Konstruktionshistorie eines CAD-Models relevant, da das CAD-Model mit dieser Historie rekonstruiert und weitergehend bearbeitet werden kann. Daher beschränkt sich diese Arbeit auf CAD-Daten, die Informationen über die Konstruktionshistorie besitzen. Informationen über die Funktion oder Festigkeit eines CAD-Models werden vorerst außen vor gelassen. Um eine Vewechslung zwischen CAD-Modellen und Large-Language-Modellen zu vermeiden, wird im Folgenden die alleinstehende Bezeichnung „Model“ vermieden.

Die von aktuellen LLM-Ansätzen demonstrierten, beindruckenden Fähigkeiten in der Verarbeitung von Text haben die Nutzung von text-basierten CAD-LLMs motiviert. Derartige CAD-LLMs arbeiten mit der text-basierten Repräsentation von CAD-Modellen, welche hier *CAD-Text* genannt wird. Der CAD-Text besteht aus durch Separatoren getrenntem Text (strings). Dieser Text lässt sich durch spezifische Schlüsselworte (Keywords) in Bausteine unterteilen, welche als Ganzes ein CAD-Modell beschreiben.

Ein Beispiel für eine CAD-Model und den dazugehörigen CAD-Text ist in der nachfolgenden Abbildung 1 zu sehen. Schlüsselworte sind unterstrichen.



```
SOL Line 0.5 -0.25 Arc 0.25 0 0 -0.25
Line 0 -1.75 Arc 0.25 -2 0.5 -1.75 Extrude 0
0 0 -1 0 0 0 0 1 2 0 NewBodyFeatureOperation
OneSideFeatureExtentType EOT SOL Line 0.0572
0.6 Line 1.4428 0.6 Line 0.75 1.8 SOL Circle
0.75 1 0.25 Extrude 0 0 -0.25 0 0 -1.25 0 1
-0.25 -1 0 CutFeatureOperation
OneSideFeatureExtentType EOT
```

Abbildung 1: Exemplarische Darstellung eines CAD-Models und dem dazugehörigen CAD-Text

Die Umwandlung von CAD-Text in CAD-Model ist an vielen Stellen erwähnt [1–8]. Die Ableitung des CAD-Texts vom CAD-Model hingegen ist kaum dokumentiert.

Dieser ausgegebene CAD-Text muss anschließend mit demselben Übersetzungsprotokoll wieder in ein CAD-Modell überführt werden, um die Aktion des LLMs innerhalb der Software zu visualisieren. Auch diese Rückübersetzung ist in der Literatur kaum dokumentiert.

Bei der Übersetzung von CAD-Text zu CAD-Modell kommt erschwerend hinzu, dass die Ausgabe des CAD-LLMs aufgrund der probabilistischen Charakteristiken der LLM-Technologie zu Fehlern im Syntax des CAD-Textes führen kann. Unter Syntax versteht man die Gesamtheit der Regeln, die zur exakten Formulierung in diesem System erforderlich sind. Syntaktische Fehler führen unweigerlich zu invaliden CAD-Modellen. Als „invalid“ werden hier Modelle bezeichnet, die geometrisch unmöglich abzubilden sind, beispielsweise, wenn eine Skizze für eine Extrusion lediglich aus zwei Linien besteht. Da hiermit keine geschlossene zweidimensionale Form erzeugt werden kann, ist die Erzeugung eines Volumenkörpers mittels Extrusion unmöglich. Um derartige Fehler zu verhindern, bedarf es eines strikten Regelwerks

für die Strukturierung von CAD-Texten. Dieses Regelwerk, welches Gesetzmäßigkeiten definiert, die bei Anwendung auf einen Text die korrekte Abbildung einer geometrischen Form sicherstellen, bezeichnen wir als *CAD-Grammatik*.

Aus diesen Überlegungen leiten sich die zentralen Forschungsfragen dieser Arbeit ab:

FF1: Wie muss eine CAD-Grammatik gestaltet sein, um die Übersetzung eines CAD-Text in komplexe CAD-Modelle durch sicherzustellen?

FF2: Wie lässt sich die Wiedergabetreue einer Grammatik-basierten Übersetzung (CAD-Modell → CAD-Text und CAD-Text → CAD-Modell) überprüfen?

Diese Arbeit unternimmt den Versuch diese Fragen mit der Vorstellung einer CAD-Grammatik und eines in eine CAD-Software integriertes CAD-Tools zu beantworten.

2. Stand der Technik

Bisherige Ansätze im Bereich der CAD-LLMs [1], [8], [9] fokussieren sich primär auf die CAD-LLM-Seite. Sie verlassen sich darauf, dass das LLM die komplexen Konstruktionsregeln implizit durch die in den CAD-Datensätzen enthaltenen Modelle lernt. Aufgrund der oft geringen Datenmenge funktioniert dieser Ansatz in der Praxis jedoch eher unzuverlässig. Definierte grammatikalische Regeln werden durch bisherige Arbeiten weder adäquat dokumentiert noch in der Anwendung zwingend forciert. Dieses unregulierte Verlassen auf die impliziten Regeln der Datensätze führt zu einer signifikanten Anzahl an invaliden CAD-Modellen [2], [7] und zu einer geringeren Kontrolle über die erzeugten Geometrien.

Weiterhin zeigen bisherige Arbeiten zwar vereinzelt eine Übersetzung von CAD-Text zu CAD-Modell, die Ableitung des CAD-Textes vom existierenden CAD-Modell bleibt unbehandelt. Eine Integration eines CAD-LLM in eine CAD-Software mit bi-direktionaler Kommunikation zwischen User und CAD-LLM ist damit nicht möglich und auch nicht dokumentiert. Zwar erwähnen einige Arbeiten eine mögliche Integration in kommerzielle Software, sie liefern hierfür jedoch weder Nachweise noch konkrete Beispiele.

3. Ansatz

Mit dem von uns entwickelten *CAD-Tool* stellen wir einen möglichen Ansatz zur Lösung dieser Probleme vor. Das CAD-Tool ist ein Interface, welches dem User direkte Interaktion mit einem CAD-LLM innerhalb einer CAD-Software ermöglicht.

Dies beinhaltet die flexible Umwandlung von CAD-Modell zu CAD-Text und zurück über die API der CAD-Software und ein sogenanntes *CAD-Dictionary*. Dieses Dictionary, vergleichbar mit dem JSON-Format der Fusion 360 Gallery [2], speichert alle notwendigen Informationen in Form von hierarchischen Key-Value-Zuordnungen. Diese Strukturierung als „Nested-Dictionary“ vereinfacht die Verarbeitung und Umwandlung von CAD-Daten.

Innerhalb dieses CAD-Dictionaries prüft das CAD-Tool kontinuierlich, ob die Gestaltungsregeln eingehalten werden. Hierfür wurden eine CAD-Grammatik sowie ein entsprechender Prüfungsalgorithmus entwickelt. Durch die Prüfung der CAD-Grammatik wird die Validität der vom CAD-LLM generierten Modelle garantiert und invalide Modelle werden ausgeschlossen.

Unterstützt wird diese Entwicklung durch die Flexibilität einer hybriden Tokenisierung [10]. Dies vereinfacht die Übersetzung des CAD-Dictionaries in die Bausteine des CAD-Texts maßgeblich.

Die flexible Umwandlung von CAD-Modell in CAD-Text und die Etablierung sowie Prüfung der CAD-Grammatik stellen die Kernbeiträge dieser Arbeit dar. Der grundlegende Workflow zwischen User, CAD-Software, CAD-Tool und CAD-LLM ist in Abbildung 2 dargestellt. Die Kernbeiträge sind dort rot umrandet.

Durch diesen Ansatz lassen sich CAD-LLMs direkt in die CAD-Software integrieren. Während dies von anderen Arbeiten nur angedeutet wurde, wird es hier zum ersten Mal praktisch dokumentiert. Das verwendete CAD-LLM lässt sich problemlos tauschen, solange dieselbe CAD-Grammatik und damit dieselbe Struktur für den CAD-Text verwendet.

Ein entscheidender Vorteil ist, dass die Qualität des CAD-LLM-Outputs, zumindest bezüglich ihrer Validität, durch die Grammatik nun erfassbar wird. Diese Messbarkeit ermöglicht es, spezialisierte CAD-LLMs gezielt durch Methoden wie Reinforcement Learning zu verbessern und Interventionen am CAD-LLM, wie Anpassungen des Trainingsdatensatzes oder der Architektur des LLMs, nicht nur mit Kenngrößen des maschinellen Lernens, sondern auch anhand von ingenieurspezifischen Kenngrößen zu bewerten.

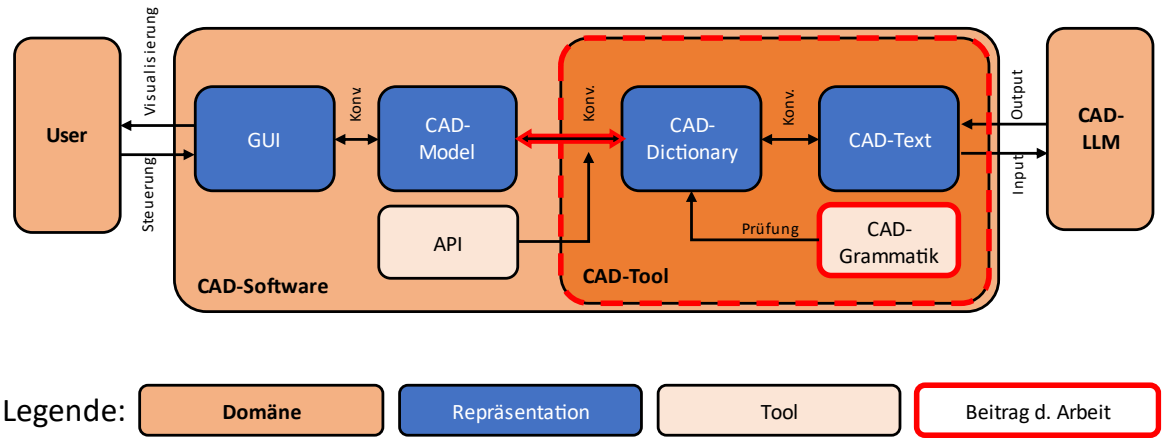


Abbildung 2: Übersicht über den Workflow von User, CAD-Software inkl. CAD-Tool und CAD-LLM

4. Anwendung

4.1. CAD-Tool

Für den in Abbildung 2 dargestellten Ablauf wurde das besagte CAD-Tool als Schnittstelle zwischen CAD-LLM und CAD-Software entwickelt. Hierfür war die API der CAD-Software notwendig, weshalb die Wahl aufgrund der extensiven API-Dokumentation auf Autodesk Inventor fiel. Konkret wurde die Version „Autodesk Inventor Professional 2024“ verwendet. Bei entsprechendem Verständnis der jeweiligen API ist der Ansatz jedoch auf jede beliebige CAD-Software übertragbar. Für die Zukunft ist die Portierung auf FreeCAD geplant, um den Zugang zum CAD-Tool zu vereinfachen.

Für eine direkte Interaktion des Users mit dem CAD-Tool wurde eine visuelle Schnittstelle direkt in die GUI der CAD-Software integriert. Dies erlaubt es dem User Prompts in Form von CAD-Text direkt an ein CAD-LLM zu senden. Dies kann ein manuelle eingegebener oder von einem CAD-Model abgeleiteter CAD-Text sein Das in dieser Arbeit genutzte CAD-LLM wurde für die sogenannte „Next-Step-Prediction“ konzipiert. Hierbei nutzt das CAD-LLM den aktuellen Zustand als Start-Prompt um den nächsten Konstruktionsschritt in Form eines CAD-Textes zu prädizieren. Das CAD-Tool erlaubt dabei in begrenztem Maße Kontrolle über das Verhalten des CAD-LLMs. Das CAD-Tool steuert den k-Wert des Top-K-Samplings, während des Generierungsprozesses des LLMs wird aus den k wahrscheinlichsten Tokens zufällig ausgewählt. Damit wird die „Kreativität“ des Modells gesteuert. Ein $k = 1$ führt zu einem deterministischen, wiederholbaren Output. Weiterhin wird die Anzahl der maximal generierten Tokens durch das CAD-Tool gesteuert. Die Schnittstelle zum CAD-Tool in der CAD-Software ist in Abbildung 3 abgebildet.

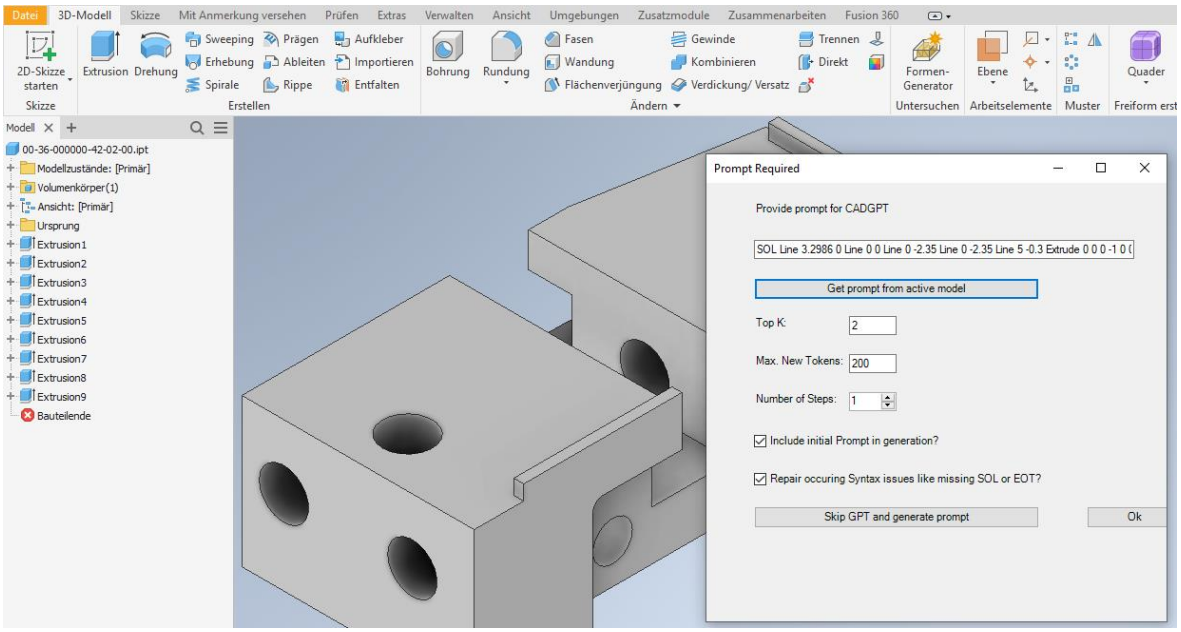


Abbildung 3: Das CAD-Tool in Anwendung in Autodesk Inventor 2024.

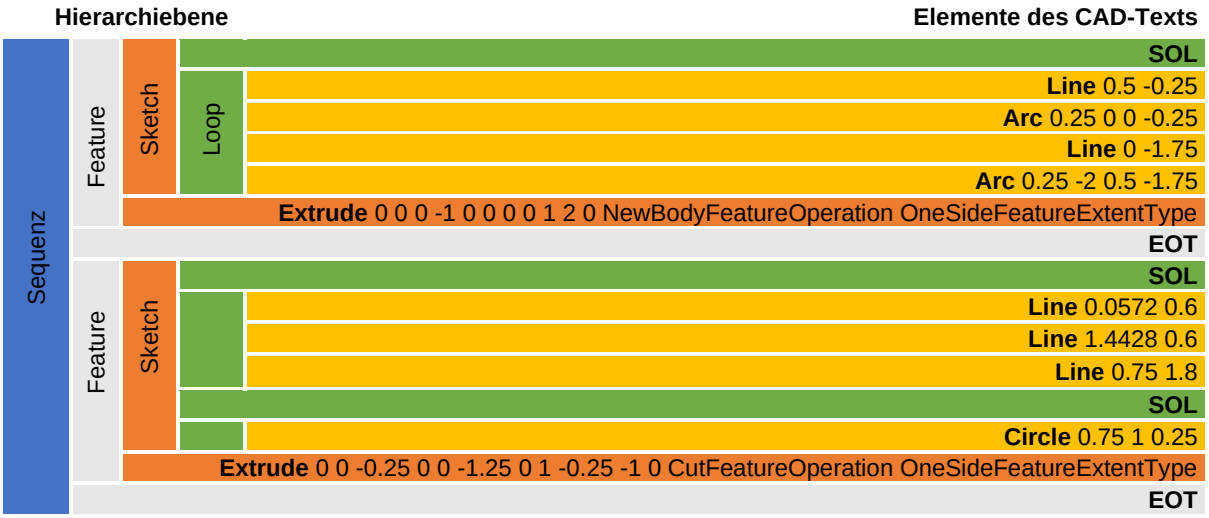
4.2. CAD-Dictionary

Als Zwischenmedium zwischen CAD-Modell und CAD-Text wird, ähnlich wie in [1], [2] und in Kapitel 3 beschrieben, ein CAD-Dictionary verwendet, da es die Speicherung und Verarbeitung der Informationen vereinfacht. Für die Übersetzung CAD-Model → CAD-Text werden Informationen aus dem CAD-Model benötigt. Um diese zu erhalten nutzt das CAD-Tool die API der CAD-Software und extrahiert die relevanten Informationen in das CAD-Dictionary. Dieses ist, ähnlich wie bei [1], [3] in verschiedene Ebenen einer Hierarchie unterteilt:

1. Das kleinste Element ist ein **Parameter**: entweder ein Zahlenwert oder ein spezifisches vordefiniertes Schlüsselwort.
2. Darauf folgt das **Sketch-Primitiv**: Eine Sammlung aus Parametern, bei der ein initiales Schlüsselwort die notwendigen nachfolgenden Parameter diktiert.
3. Die nächste Ebene ist der **Loop**: Eine Sammlung von Sketch-Primitiven, die eine geschlossene Skizze bilden.
4. Darauf folgt die **Sketch**-Ebene: Eine Sammlung von Loops zu einer vollständigen Skizze.
5. Das **Feature**: Definiert durch 3D-Operationen (z.B. Extrude oder Rotation), die durch eine Reihe von Parametern charakterisiert sind.
6. Die höchste Ebene ist die **Sequenz**: Eine Sammlung von Features, die das gesamte Bauteil repräsentiert.

Loops werden durch den speziellen Parameter *SOL* (Start of Loop) und verschiedene Features durch *EOT* (End of Text). Die Hierarchie für das in Abbildung 1 angeführte Beispiel sieht dementsprechend folgendermaßen aus:

Tabelle 1: Einordnung der Elemente eines CAD-Textes in die Hierarchieebenen



Bei der Übersetzung CAD-Model → CAD-Text wird ausgehend von diesem Dictionary wird der CAD-Text generiert und kann an das CAD-LLM gesendet werden. Wir setzen hierbei voraus, dass existierende, in der CAD-Software korrekt geplottete CAD-Modelle syntaktisch valide sind und die Übersetzung in CAD-Text fehlerfrei abläuft.

Bei der umgekehrten Übersetzung (CAD-Text → CAD-Modell) wandelt das CAD-Tool einen CAD-Text in das CAD-Dictionary um. Der Ursprung dieses CAD-Textes ist dabei in aller Regel ein entsprechend trainiertes CAD-LLM. Bei dieser Übersetzung wird auf Grundlage des CAD-Dictionaries die CAD-Grammatik aller Elemente aller Hierarchieebenen geprüft.

Ist die Prüfung erfolgt, nutzt das CAD-Tool erneut die API um alle validen Elemente des CAD-Dictionaries in ein CAD-Modell umzuwandeln und dem User in der GUI zu präsentieren.

4.3. CAD-Grammatik

Die CAD-Grammatik wurde für alle oben beschriebenen Ebenen entwickelt. Dabei wurden zunächst Grundregeln in folgenden Aspekten definiert:

1. einfachste geometrische Prinzipien (bspw. eine Fläche kann nicht aus lediglich zwei Linien bestehen)
2. die oben beschriebene Hierarchie (Bsp.: nach einem SOL Element muss ein Sketch-Primitiv folgen),
3. der Daten-Typ der Parameter (Bsp.: nach einem „Line“ Element müssen zwei Parameter des float-Typen folgen)
4. zulässige Schlüsselworte (Bsp.: Circle ist valide, Circe nicht).

Elementarer Bestandteil der Entwicklung der Grammatik war der zweite Schritt, bei dem direkte Experimente mit einem CAD-LLM durchgeführt wurden um die CAD-Grammatik „in situ“ erprobt werden konnte. Mit Hilfe dieser Experimente konnte das Regelwerk erweitert werden, da das CAD-LLM Fehler verursachte, die während der initialen Entwicklung nicht berücksichtigt wurden. Tabelle 2 zeigt weitere Beispielregeln und dazugehörige Fehler. Mit einem Asterisk (*) markierte Regeln können durch das CAD-Tool bei einem Fehler korrigiert werden Es ist zu beachten, dass diese Tabelle und die o.g. Beispiele nicht vollständig sind.

Tabelle 2: Beispielhafte Regeln der CAD-Grammatik.

Parameter	Beispielfehler
Als String sind ausschließlich definierte Schlüsselworte valide: Line, Arc, Circle, Extrude, SOL, EOT* „-0“, „0.0.“, „-0.0“ sind als Darstellungen von „0“ nicht valide* …	„Linie“; „ArcArcArc“ „… Line -0 -0 Arc …“
Sketch-Primitiv	**Beispielfehler**
Auf jeden Linien-Befehl müssen zwei Zahlen folgen Der Radius des Kreises darf einen Mindestwert nicht unterschreiten …	„… Line 0.15 Line …“ „Circle 0 0 0.0000005“
Loop	**Beispielfehler**
Ein valider Sketch besteht mindestens aus drei (validen) Linien, aus einer (validen) Linie und einem (validen) Arc oder aus einem (validen) Kreis Innerhalb eines Loops können nicht zwei identische Linien liegen …	„…SOL Arc […] SOL…“ „…Line 0 0 Line 0 0…“
Sketch	**Beispielfehler**
Nach jedem SOL folgt zwingend ein Sketch-Primitiv* Ein Sketch muss mindestens einen validen Loop enthalten …	„…EOT SOL Extrude…“ „SOL Line […] Extrude…“
Feature	**Beispielfehler**
Ein Extrude-Feature enthält 15 Parameter mit definierten Datentypen: 1. Position: [string] „Extrude“, 2.-10. Position: [floats] XYZ Koordinaten für drei Ebenen-Punkte 11.-12. Position: [float] Ausdehnung in positive und negative Richtung. 13.-15. Position: [string] Extrude Typ, vordefiniert durch API XYZ Koordinaten müssen eine eindeutige Ebene erstellen …	„…Extrude 0 0 0 EOT“ „Extr 0 0 0 1 0 0 0 0 1 …“ „Extrude Arc 0 0 Arc 0 …“ „…1 0.5 SOL New…“ „ThreeSideFeatureExtension“ „Extrude 0 0 0 0 0 0 0 0 1“
Sequenz	**Beispielfehler**
Nach jedem Extrude steht ein EOT* Nach jedem EOT folgt ein SOL* Die Sequenz muss mindestens ein valides Feature enthalten …	„Line[…] Extrude […] SOL…“ „…EOT Line […]“ [zu viele Fehler in Features]

4.4. Prüfung der CAD-Grammatik

Diese Regeln werden für alle Elemente aller Ebenen geprüft. Werden alle Regeln eines Elements eingehalten, wird das Element im CAD-Dictionary als „valide“ markiert und für weitere Schritte berücksichtigt. Wird eine Regel verletzt, wird dieser Verstoß im Dictionary für spätere Auswertungen dokumentiert. Dabei kann im Härtegrad variiert werden: 1) bei fatalen Fehlern die zu einer invaliden Geometrie führen, wird ein „Error“ vermerkt; 2) „Schönheitsfehler“, wie sehr kleine Linien, die zwar eine valide Geometrie erzeugen aber als „unsauberes Arbeiten“ gelten, erhalten lediglich ein „Warning“. Diese Variationen in der Fehlerbehandlung lassen zukünftigen Auswertungsalgorithmen viel Flexibilität. Im aktuellen CAD-Tool werden jedoch beide Varianten jedoch vorerst gleich behandelt. In vereinzelten Fehler-Fällen kann das CAD-Tool eine Korrektur des Regelverstoßes durchführen. Ist die Reparatur erfolgreich, wird das Element nachträglich als „valide“ markiert. Ist eine Reparatur nicht möglich, wird das Element als „fehlerhaft“ deklariert und für die nächsten Prozessschritte ignoriert (siehe Abbildung 4).

Auf diese Weise wird die Einhaltung der CAD-Grammatik lückenlos dokumentiert und auswertbar gemacht.

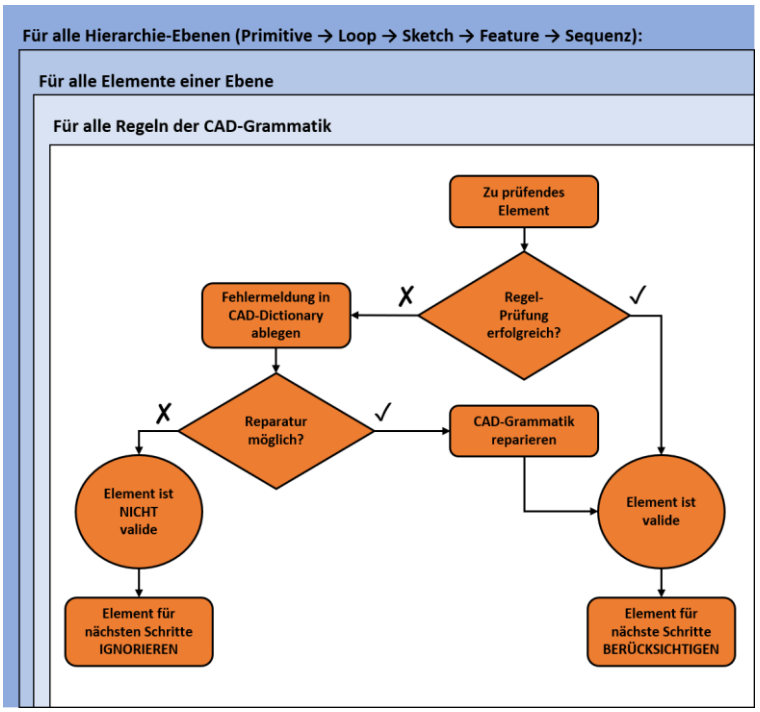


Abbildung 4: Überprüfungsprozess der CAD-Grammatik

5. Ergebnisse

Das von dem CAD-Tool angesteuerte CAD-LLM ist Teil eines aktuellen Forschungsvorhabens und befindet sich noch in einer frühen Entwicklungsphase. Die in Kapitel 3 angedeutete hybride Tokenisierung ist Kernbestandteil der Erforschung dieses LLMs. Bedingt durch die geringe Verfügbarkeit von Trainingsdaten besitzt das CAD-LLM verhältnismäßig wenige Parameter [11] und tendiert entsprechend häufig zur Generierung von invalidem CAD-Text.

Gerade im Umgang mit diesem fehleranfälligen CAD-LLM demonstriert die entwickelte CAD-Grammatik ihre Wirkung: Syntaktisch fehlerhafte Ausgaben des CAD-LLMs können zuverlässig entdeckt und im CAD-Dictionary dokumentiert werden (**FF1**). Dabei demonstriert der Ansatz bereits in der Entwicklung der CAD-Grammatik seine Bedeutung als Interface zwischen User und CAD-LLM: Durch Experimente mit dem CAD-LLM konnten schnell weiterer Regeln ermittelt werden.

Das CAD-Tool kann kleinere Fehler automatisch reparieren, sofern die Reihenfolge logisch eindeutig ist (Beispiel: Fehlt ein SOL-Token nach einem EOT-Token, wird es automatisch eingefügt). In schwerwiegenderen Fällen müssen defekte Primitive oder ganze Features weggelassen werden, um die Generierung eines komplett invaliden Modells zu verhindern. Dadurch wird ein korrekter Syntax erzwungen, die ein valides CAD-Model repräsentiert und von der CAD-Software überhaupt erst verarbeitet werden kann (**FF1**).

Ein wesentliches Qualitätsmerkmal des CAD-Tools liegt in der Wiedergabetreue. Eine vollständige Hin- und Rückübersetzung (CAD-Modell → CAD-Text → CAD-Modell → CAD-Text) mit dem CAD-Tool liefert identische CAD-Texte und CAD-Modelle. Minimale Abweichungen treten zwar auf, sind aber auf Rundungen der Dezimalstellen zurückzuführen. Während des Zugriffes auf das CAD-Model werden alle Werte auf die vierte Nachkommastelle gerundet. Abweichungen vom Ursprungsmodel betragen damit weniger als $1 * 10^{-4}\mathrm{mm}$ und sind realistisch zu vernachlässigen. Diese hohe Wiedergabetreue wurde erfolgreich auf einer Reihe von Beispielgeometrien getestet (**FF2**).

Trotz dieses Machbarkeitsnachweises besitze der Ansatz Limitationen die in Zukunft behandelt werden müssen. So beschränkt sich die CAD-Grammatik momentan primär auf die geometrische „Machbarkeit“ eines CAD-Modells. Aspekte wie Zwangsbedingungen

(Constraints) und Bauteilfunktionen werden weitestgehend ignoriert. Damit prüft die CAD-Grammatik nur einen kleinen Teil aller Aspekte eines CAD-Modells.

Weiterhin Verletzungen der CAD-Grammatik sehr rigoros beantwortet, indem invalide und irreparable Elemente für die CAD-Model-Erzeugung schlicht ignoriert werden. Abgesehen von einem Vermerk im CAD-Dictionary findet keinerlei weiterführende Fehlerkommunikation statt. Damit kann ein CAD-Model unvollständig generiert werden, ohne dass der User etwas davon erfährt.

Die Verwendung eines sequentiellen CAD-Texts bringt zudem den Nachteil mit sich, dass Referenzen zwischen verschiedenen Elementen einer Konstruktion, also beispielsweise zwei Extrusionen, nicht abgebildet werden können. Damit entfallen vorerst alle Operationen, bei denen ein existierendes Element referenziert wird. Dazu zählen beispielsweise Fasen, Rundungen, aber auch „Bis zu“-Tiefenangaben bei Extrusionen.

6. Zusammenfassung

Diese Arbeit stellt eine CAD-Grammatik vor, welche während der Übersetzung eines von CAD-LLM generiertem CAD-Text in CAD-Modelle die syntaktische Validität des CAD-Textes prüft. Die CAD-Grammatik wurde über ein CAD-Tool in eine CAD-Software (Autodesk Inventor) integriert. Diese CAD-Tool erlaubt die direkte Steuerung eines CAD-LLMs innerhalb einer CAD-Software. Durch die Verwendung der CAD-Grammatik kann die Anzahl generierten invaliden Geometrien kann drastisch gesenkt werden. Die strukturiert erfassten Informationen über Verletzungen der CAD-Grammatik, also syntaktische Fehler im CAD-Text können als wertvolles Feedback für die Verbesserung eines CAD-LLMs genutzt werden, beispielswiese mittels Reinforcement Learning. Dadurch kann dessen Performance ohne zusätzliche Datensätze zu gesteigert werden.

Ein logischer nächster Schritt ist die Erweiterung der CAD-Grammatik um bisher nicht unterstützte Operationen und Schlüsselwörter sowie die Integration von loseren, heuristik- und erfahrungsbasierten Empfehlungen (Konstruktions-„Standards“).

So kann die CAD-Grammatik ein maßgeblicher Faktor für die gezielte Erzeugung synthetischer, aber syntaktisch garantierter Datensätze sein. Diese Anwendung würde die Datengrundlage und damit die Leistungsfähigkeit von CAD-LLMs stärken.

Literaturverzeichnis

[1] Wu, Rundi ; Xiao, Chang ; Zheng, Changxi: DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. DOI: 10.1109/ICCV48922.2021.00670

[2] Willis, Karl D. D. ; Pu, Yewen ; Luo, Jieliang ; Chu, Hang ; Du, Tao ; Lambourne, Joseph G. ; Solar-Lezama, Armando ; Matusik, Wojciech: Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences, arXiv (2021). DOI: 10.1145/3450626.3459818

[3] Zhang, Zhanwei ; Sun, Shizhao ; Wang, Wenxiao ; Cai, Deng ; Bian, Jiang: FlexCAD: Unified and Versatile Controllable CAD Generation with Fine-tuned Large Language Models, arXiv (2024). DOI: 10.48550/arXiv.2411.05823

[4] Ganin, Yaroslav ; Bartunov, Sergey ; Li, Yujia ; Keller, Ethan ; Saliceti, Stefano: CADL: Computer-Aided Design as Language, arXiv (2021). DOI: 10.48550/arXiv.2105.02769

[5] Dupont, Elona ; Cherenkova, Kseniya ; Mallis, Dimitrios ; Gusev, Gleb ; Kacem, Anis ; Aouada, Djamila: TransCAD: A Hierarchical Transformer for CAD Sequence Inference from Point Clouds, arXiv (2024). DOI: 10.1007/978-3-031-73030-6_2

[6] Yu, Nomi ; Alam, Md Ferdous ; Hart, A. John ; Ahmed, Faez: SynthBal: GenCAD-3D: CAD Program Generation using Multimodal Latent Space Alignment and Synthetic Dataset Balancing. In: *Journal of Mechanical Design* Bd. 148 (2026), Nr. 3, S. 031703. DOI: 10.1115/1.4069276

[7] Xu, Xiang ; Willis, Karl D. D. ; Lambourne, Joseph G. ; Cheng, Chin-Yi ; Jayaraman, Pradeep Kumar ; Furukawa, Yasutaka: SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled Codebooks, arXiv (2022). DOI: 10.48550/arXiv.2207.04632

[8] Willis, Karl D. D. ; Jayaraman, Pradeep Kumar ; Lambourne, Joseph G. ; Chu, Hang ; Pu, Yewen: Engineering Sketch Generation for Computer-Aided Design, arXiv (2021). DOI: 10.1109/CVPRW53098.2021.00239

[9] Seff, Ari ; Ovadia, Yaniv ; Zhou, Wenda ; Adams, Ryan P.: SketchGraphs: A Large-Scale Dataset for Modeling Relational Geometry in Computer-Aided Design, arXiv (2020). DOI: 10.48550/arXiv.2007.08506

[10] HABIBI, SAYEDA HADISA ; BERGELT, JULIA ; TEICHMANN, MICHAEL ; HAMKER, FRED H.: *Influence of tokenization strategies on the prediction of CAD model descriptions*. Bd. 2026. DOI: https://doi.org/10.17619/UNIPB/1-2654

[11] KAPLAN, JARED ; MCCANDLISH, SAM ; HENIGHAN, TOM ; BROWN, TOM B. ; CHESS, BENJAMIN ; CHILD, REWON ; GRAY, SCOTT ; RADFORD, ALEC ; U. A.: Scaling Laws for Neural Language Models, arXiv (2020). DOI: 10.48550/arXiv.2001.08361