

# איך שולחים 2,500 PRs בחודש עם agents: Lauren Tan Matt Pocock

Verification, סביבה מוגבלת ולולאה פנימית וחיצונית: המעבר מבישול ביתי למטבח מישלן  
של agents

Lauren Tan (poteto), Matt Pocock · שידור חי, המקור המקורי לא זוהה (הועלה מחדש ב-X על ידי Morlex) · 2026-  
10-03

משך: כ-63 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

## על המסמך הזה

זה שידור חי של כשעה. Matt Pocock (Total TypeScript, מחבר ספריית skills פופולרית) מראיין את Lauren, שמופיעה על המסך בכינוי poteto. היא עבדה בצוות React ב-Meta וב-Netflix, וכיום עובדת ב-Cursor, שלדבריה הפכה ל-SpaceX AI. השיחה בנויה כ-Q&A על הרצאה קודמת שלה, "how I shipped 2,500 PRs last month to production". לפי Matt, ההרצאה צברה ב-X כ-3 מיליון צפיות. בכיתוב של הפוסט ב-X כתוב "1000 of PRs a month", אבל בתמליל המספר הוא 2,500 בחודש האחרון.

הטענה המרכזית שלה: כשהמודלים טובים מספיק, צוואר הבקבוק עובר מה-agent אל הסביבה שבה הוא עובד. אמון ב-agent נבנה קודם כול דרך verification, כלומר יכולת של ה-agent להריץ את הקוד ולראות את התוצאה. אחר כך מגיעים קוד מוגבל עם lint rules ו-conventions, ואז חיבור של לולאה חיצונית (Slack, Linear, X) ללולאה הפנימית של ה-agents. בסוף ה-review הופך לדגימה ולא לחסימה. את כל זה היא מתארת במטאפורה של מטבח מישלן במקום software factory.

### איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, וגם מה שלא נאמר.
- 2. **סולם האמון:** איך הגיעה ל-skills, ולמה מומחיות תחום ושפה חשובות עכשיו יותר.
- 3. **מטבח מישלן:** המטאפורה שמחליפה את ה-software factory.
- 4. **Verification:** ה-skill החשוב ביותר, ולמה החלק הדטרמיניסטי עובר ל-CLI.
- 5. **הסביבה:** lint rules, ה-framework הפנימי Dune ופירוק god files.
- 6. **2,500 PRs:** לולאה פנימית וחיצונית, coordinator agents ו-gardening.
- 7. **Review ו-dark factory:** דגימה, full autopilot ומה עושים עם one-way doors.
- 8. **Skills כתהליך:** כריית transcripts, ה-skill בשם Recall, ושילוב ספריות skills.
- 9-10: **צעדים מעשיים,** שאלות פתוחות ומילון מונחים.

**הערת מקור** המסמך הופק אוטומטית מתמליל ההקלטה ומהפריימים שלה. זו שיחת וידאו של שני משתתפים בלי שקפים, ולכן מוצגת רק תמונה אחת לזיהוי הדוברים. התמלול האוטומטי (ASR) עלול לשבש שמות ומונחים: שם המשפחה Tan מופיע רק בכיתוב של הפוסט ב-X, בתמליל היא Lauren או Potato ועל המסך poteto. ה-framework של ה-skills שלה מופיע כ-P-Stack, P-Stack ו-PSAC. ה-agent מופיע כ-Groftbot, Groftbot, GrokBot, והשם המדויק לא ודאי. הציטוטים הם מילה במילה מהתמליל ונבדקו מול חותמת הזמן. לא נוסף מידע ממקורות חיצוניים.



מנכ"ש השידור: Matt Pocock משמאל ו-Lauren (בכינוי poteto) מימין. שיחת וידאו בלי שקפים.

# 1. תקציר מנהלים

השיחה לא עוסקת ב-prompt טוב יותר או במודל טוב יותר. היא עוסקת בהשקעה מכוונת במה שמסביב ל-agent: כלים שמאפשרים לו לבדוק את עצמו, קוד שמאפשר רק דרך אחת לעשות דברים, וצינורות שמביאים אליו context בלי שהמהנדס יעביר אותו ידנית. ה-PRs 2,500 הם תוצאה של ההשקעה הזו, ולא של פתיחה ידנית של 2,500 שיחות.

- **Verification הוא ה-skill הראשון שצריך לבנות:** לדבריה זה ה-skill הראשון שבנתה ב-Cursor, וזה שאפשר לה לטפס בסולם האמון. ה-agent מקבל ידיים ועיניים: הוא מריץ את האפליקציה ולוקח traces ו-14:03 snapshots [15:40]. היום לכל אפליקציה ב-Cursor יש verification skill שמתוחזק אוטומטית [17:17].
- **את החלק הדטרמיניסטי מוציאים לקוד:** ה-skill הוא wrapper דק סביב CLI שעובד מול Playwright ו-Chrome DevTools Protocol. בלי ה-CLI, כל agent בנה מחדש סקריפט בדיקה משלו, וכל אחד בדרך אחרת [20:26].
- **העבודה החדשה של המהנדס היא הסביבה:** lint rules מגבילים, תיקייה נפרדת לכל feature, ו-framework פנימי בשם Dune שהיא מתארת כ-Next.js פנימי לאפליקציות [28:11]. Electron. כל טעות חוזרת של agent הופכת ל-lint rule [29:42].
- **את ה-PRs 2,500 היא לא פותחת ידנית:** GrokBot (השם לא ודאי ב-ASR) מנטר את X, Slack, המייל ו-Linear ומשמש כ-outer loop. הוא שולח עבודה ל-Cursor Projects, שהם coordinator agents בענן שמחלקים אותה ל-sub-agents [40:37–42:11].
- **רוב ה-PRs הם gardening ולא agents features:** סורקים את הקוד ומחפשים דפוסים רעים. חלקם רושמים ממצאים למסמך שממש buffer במקום לתקן מיד, כדי שיהיה אפשר לזהות דפוס משותף [43:50–45:21].
- **ה-review הופך לדגימה:** היא סוקרת PRs אחרי שכבר נכנסו, דרך ה-commit history בבוקר. מצב full autopilot ב-pstack מפעיל verifier agents (עד כ-10 ל-PR) שעושים fuzzing לאפליקציה [50:06–51:37].
- **מומחיות תחום שווה היום יותר, לא פחות:** צוואר הבקבוק הוא העברת הכוונה שלך ל-agent. מילה מדויקת אחת, כמו tautological tests או TDD, דוחסת הרבה [6:22–7:55]. intent.
- **בלי verification אין autopilot:** על one-way doors בתחומים כמו רפואה ופיננסים היא אומרת בגלוי שאין לה תשובה, ומצביעה על שפות כמו Bend שמשלבות קוד עם proofs [54:42].

## מה לא נאמר

- **אין נתוני איכות:** לא נמסרו שיעור revert, מספר רגרסיות ב-production, או כמה מה-PRs הם features וכמה gardening.
- **אין נתוני עלות:** על full autopilot נאמר רק "[50:06] 'quite token intensive', בלי מספרים.
- **חוויית הצוות לא נידונה:** Matt שאל "[37:33] 'your team must hate you', והתשובה עברה ל-sampling ולא לשאלה איך עמיתים אנושיים מתמודדים עם הנפח.
- **אין הדגמה:** לא הוצגו קוד, demo או שקפים. Dune לא קוד פתוח, ו-GrokBot מתואר רק במילים.
- **טענות שנאמרו בעל-פה בלבד:** ש-Cursor היא עכשיו SpaceX AI, והשימוש ב-Opus 5.5. המסמך לא מאמת אותן.

# 2. סולם האמון: מ-proxy skills-

הסיפור מתחיל לפני Cursor. אחרי Meta, שם עבדה בצוות React, היא לקחה חודש חופש בגלל שחיקה ופתחה פרויקט צד. בינאר או פברואר כולם דיברו על orchestration ב-TUI. היא שמה לב שהיא מבזבזת שעות על micromanagement של agent יחיד, ושקשה למדוד את ההשפעה של skill. בלשונה: "[1:37] flying blind". הפרויקט עדיין פתוח ב-GitHub בשם noodle (לפי ה-ASR: potato/noodle), עם skills ותיקיית brain. המטרה הייתה לחלץ את היכולת שלה ולהעביר אותה ל-agent [3:12].

היא הצטרפה ל-Cursor במרץ. בתחילת אפריל עבדה על ביצועי ה-flame graphs ו-agent window. snapshots, הכול ידני. בהתחלה זנחה את ה-skills האישיים, ואז גילתה שהלקחים שלהם, ובעיקר verification ועבודה קפדנית, רלוונטיים מאוד. גם agents מה-frontier נוטים לקצר דרך, ולכן לדבריה "a lot of the skills that i built have been around how do i make the easy thing the right thing" [4:48].

— Lauren Tan [3:12] I was sort of the meat proxy in a way. I was the meat proxy between my agent and Chrome DevTools

בשאלה אם מומחיות תחום מאבדת מערכה, היא חושבת שההפך נכון. כשהמודלים טובים, צוואר הבקבוק הוא "your ability to express your intent and your goals in a clear way that the agent can understand and actually carry out" [6:22]. לכן רופא או עורך דין עם קצת סקרנות טכנולוגית יכולים לבנות מוצרים טובים. Matt מוסיף שהוא אובססיבי לשפה: הוא מחפש את המילה שה-agent נתפס בה וממחזר אותה ב-thinking traces שלו. TDD היה דוגמה מוקדמת, והוויכוח אם לעבוד ב-TDD עם agents פחות חשוב מהעובדה שהמילה משנה את סדר העדיפויות של ה-Lauren [7:55]. agent אימצה ממנו את הטיפ על tautological tests: מילה אחת שדוחסת הרבה intent נגד טסטים חסרי ערך.

### 3. מטבח מישלן במקום software factory

היא לא אוהבת את המונח software factory. הוא לא שגוי, אבל המילה factory לא מתקשרת לאיכות ולאומנות [9:27]. המטאפורה שלה היא מטבח מישלן, ובמקביל היא משמשת אותה גם לתאר את סולם האמון. טבח ביתי עושה הכול לבד: חיתוך, הכנות וניקיון. כשכל המשפחה נכנסת למטבח נוצר כאוס, כי אף אחד לא יודע איפה הכלים (Matt: "max capacity of one person") [11:00–12:30]. שף כבר לא מבשל הכול בעצמו. הוא כמו tech lead או CEO של המטבח: מחליט מתי להזמין מצרכים, איך לאחסן ומתי להכין. האחריות לתוצאה ולמוניטין נשארת שלו. ובלשונה: "how you set up your kitchen right and how you set up your skills your environment your code base i" [12:30]. think are ultimately the new ingredients that go into building product"

המטאפורה חוזרת לאורך כל השיחה. סכינים קהות הן מצב של מפתח בלי כלים, למשל מי שמכיר רק Notepad ולא שמע על Git. מועכי שום מתאימים לעבודה הדטרמיניסטית [26:37–25:03]. ה-scaling הוא פתיחה של רשת מסעדות, כמו Gordon Ramsay שלימד את ה-executive chef את כל הטריקים, וכל שיחה גדולה היא מסעדה שהיא עוברת בין סניפיה, "helicoptering between them" [32:49].

### 4. Verification — ה-skill החשוב ביותר

Verification פירושו לתת ל-agent ידיים ועיניים: הוא מריץ את הקוד, מתקשר איתו כמו משתמש, עושה debug ולוקח traces ו-skills [15:40]. snapshots אחרים שבנתה, כמו how, why ו-unslob, לא עזרו לה לטפס בסולם האמון, כי היא עדיין הייתה המתווכת בין ה-agent לתוצאה. לדבריה, verification הוא מה שהופך loop ללולאה אמיתית. מקרה השימוש הראשון היה עבודת הביצועים על ה-agents window. המטרה הייתה hill climbing: rubric שנותן ציון, ו-agent שמשפר בלי הפסקה מול הציון הזה. היא מזכירה גם את auto research של Andrej Karpathy (ב-ASR: Carpathy) [17:17]. כיום לכל אפליקציה ב-Cursor/SpaceX AI יש verification skill שמתוחזק אוטומטית, והוא הפך ל-critical infrastructure של הצוות.

Lauren Tan [14:03] I think that the single most important skill that should be in your toolkit is — verification.

Matt שאל על ה-CLI המותאם שבנתה. הרעיון נולד בתקופה שבה context window היה הדאגה המרכזית, כי compaction עוד לא עבד טוב. היום היא מסבירה אותו אחרת: עבודה היא רצף בין חלקים שדורשים שיקול דעת לבין חלקים מכניים, ואת המכני צריך להוציא לקוד. "I try to extract out the deterministic parts and turn that into" [20:26]. "code and just leave only the parts that actually require judgment to the agent" [20:26]. הוא רק glue שעובד מול Playwright ו-Chrome DevTools Protocol. הטריגר היה תצפית: בלי CLI, ה-agent "would basically rebuild the world each time. And then every agent did it differently" [20:26]. הייתה בזבוז של context ושל זמן. אותו עיקרון חל על מיגרציות: codemods ו-AST transforms עושים את העבודה המכנית במקום ה-agent [21:58–23:28].

### 5. הסביבה היא העבודה

Matt נותן הגדרה: "a good code base is a code base that's easy to make changes in" [23:28]. כלומר הרבה guardrails ומסלולים צרים. Lauren הולכת רחוק יותר: "I almost feel like the new job of the engineer is really to spend time on the environment" [23:28]. מי שלא השקיע באמון ובכלים תקוע בתחתית הסולם, עושה micromanagement ל-agents, ואין לו זמן לשפר את התהליך [25:03].

ההשראה שלה מגיעה מ-TypeScript, ובעיקר מ-type narrowing: מכווצים את מרחב האפשרויות עד שנשאר טיפוס אחד [28:11–26:37]. Dune, ה-framework הפנימי שהיא בונה, לא קוד פתוח. הוא מגיע עם lint rules מגבילים מאוד, כל feature בתיקיה משלו, ו-registry שמגלה את ה-features בסריקה של הקוד. "the code base is designed in a way that there's really only one way to do something". הטריגר: הגרסאות הראשונות של GrokBot היו בנויות משמונה god files, כל אחד באורך של לפחות 10,000 שורות [29:42]. Matt מסכם שהיתרון של כללים בסביבה הוא שה-agent לא צריך לזכור אותם. הוא נתקל בהם ברגע הנכון [31:14].

— **Lauren Tan [29:42]** every time you see a mistake, every time you see something that could be done better, you step back and think, how do I turn this into a lint rule? How do I make it so that the code base makes this impossible, right

## 6. איך מגיעים ל-2,500 PRs: לולאה פנימית ולולאה חיצונית

התנאי המוקדם הוא הסביבה מהפרק הקודם. בלעדיה, לדבריה, הנפח הזה לא היה אפשרי [31:14]. מבחינתה הלולאה הפנימית היא agents שעובדים לקראת snapshot של הכוונה שלה, וה-snapshot הזה מתיישן. דיווחי באגים ובקשות מגיעים ל-Slack, ל-Linear ול-X, ובלי טריגרים היא נאלצת להעביר את ה-context ידנית [32:49–34:25]. החיבור בין הלולאות הוא המנוף: ערוץ Slack שה-agent מוני עליו (דרך Slack MCP או subscription harness-), טריאז' אוטומטי, שחזור הבאג על main בעזרת ה-verification skill, ובדיקה שהבעיה לא נובעת מה-setup של המשתמש. השאלה הקבועה שלה: "where am I the bottleneck in this process? Why do my agents need me to answer this question?" [35:59]. ה-unlock המרכזי לדבריה הוא לחשוב מהסוף להתחלה: מה צריך כדי שה-agent ימזג את הקוד של עצמו [37:33].

בפועל, GrokBot (שם לא ודאי) הוא personal agent עם connectors למייל, ליומן, ל-Slack, ל-X ול-Linear, עם routines שמנטרים אותם [40:37–39:05]. הוא שולח משימות ל-Cursor Projects. coordinator הוא agent בענן, עם מחשב משלו, שלא עושה עבודה בעצמו אלא מחלק אותה ל-sub-agents. הדוגמה שלה: 30 issues של ביצועים שמגיעים בבת אחת. אפשר להפעיל agent לכל issue, אבל אז הקשר ביניהם הולך לאיבוד, עבודה מתכפלת, ושורש הבעיה, שנמצא אולי במקום אחר, נשאר מוסתר [42:11]. מ-Netflix היא מביאה את העיקרון "context, not control" [39:05].

— **Lauren Tan [42:11]** and cursor projects are my inner loop and Grokbot is my outer loop. Grokbot takes all the context, external context, gives it to the projects

מקור העבודה השני הוא הקוד עצמו. לדבריה [43:50] "a lot of the work actually is spent on gardening", כלומר שיפור הסביבה לטובת כל הצוות, כך שגם עובד חדש יהיה פרודוקטיבי מהיום הראשון. דוגמה: routine שמחפש footguns של React. היא מסבירה: "i don't actually tell it to fix the issue first i tell it to append it to a document" [45:21]. כל כמה ימים היא עוברת על המסמך ומזהה שכמה ממצאים הם בעצם אותה בעיה. ה-buffer מכריח הסתכלות על התמונה הגדולה, וזה גם התפקיד של ה-chief of staff agent: לראות את היער ולא רק לבצע [46:54].

## 7. Review, dark factory ו-one-way doors

על השאלה אם טעמה את כל 2,500 המנות: "you cannot be tasting every single dish that comes out of your kitchen especially if you have multiple restaurants so it becomes more about sampling" [46:54]. היא דוגמת PRs, בוחנת אותם בקפדנות ומתקנת את הסביבה ולא את ה-agent הבודד. אירוע חד-פעמי לא מצריך תיקון. כשכמה agents לוקחים אותו קיצור דרך, צריך lint rule, constraint או skill [48:28]. היא מזכירה: "i don't want to sell this as something that you can just do easily by using PSAC" [48:28].

Matt הציע שזו לא dark factory, והיא תיקנה: במובן מסוים כן. ה-agents ממזגים בזמן שהיא ישנה. יש לה יותר מעשרה chief of staff agents, כל אחד על תחום: ביצועים של אפליקציית ה-desktop של GrokBot, באגים שמשתמשים מדווחים, וכתיבה מחדש ניסיונית כאפליקציה [50:06] "native, just a toy". מצב full autopilot ב-pstack מפעיל verifier agents לכל PR. הם עושים fuzzing לאפליקציה, מחפשים רגרסיות ומתקנים עד שה-PR מוכן להיכנס. אפשר לכוון ולהוריד מ-10 verifiers לאחד. בבוקר היא עוברת על ה-commit history, ואם צריך עושה revert או מוסיפה lint rules. היום הראשון היה מפחיד: "i think what if I cause a sev" [51:37]. Matt: "i think the idea of sampling instead of blocking i think is really important" [53:10].

Matt שאל על סביבות רגישות כמו רפואה, משפט ופיננסים, ועל PRs שהם one-way doors, למשל כאלה שגורמים לאובדן נתונים. התשובה שלה: הכול תלוי באיכות ה-verifiability. "aspect to be able to do this" [54:42]. בתחום שקל לאמת, one-way doors הופכים בפועל ל-two-way doors. תוכנה ומתמטיקה הן דוגמאות לתחומים כאלה. היא מודה: "I think it's a great question that I don't really have" [54:42]. the answer to. כתקווה לעתיד היא מזכירה את Bend, שפה שמשלבת קוד עם proofs, במקום לכתוב proofs בשפה נפרדת כמו Lean או TLA+ [56:13].

## 8. Skills הם תהליך

Matt רואה ב-skills תהליך שתורגם למילים, ושאל איך משלבים את pstack עם הספרייה שלו. Lauren רואה בהן ספריות משלימות, ומדגישה שלכל אחד צריך להיות סט סכינים משלו. אפשר לשלב את Grill Me או Wayfinder של Matt (כך לפי ה-ASR) עם skills של ביצוע מ-pstack, או ליצור mode חדש מ-Wayfinder ומ-potato mode [59:21–1:00:53]. הטיפ המרכזי, שהיא מייחסת ל-Matt: לזכור את ה-transcripts הקודמים, למצוא את הרגעים שבהם התערבת ותיקנת את ה-agent, ולהפוך אותם ל-skill או ל-lint rule. היא מסבירה: "it's the process" [59:21]. materialized. It's the real process, not an abstract idea in your head

כך נולד ה-skill שלה בשם Recall. בעבודה על virtualization ב-Cursor היו הרבה באגים, ובכל שיחה חדשה היא רצתה להעביר context מהשיחה הקודמת בלי לכתוב חיבור ארוך. ההערכה שלה למגמה: ב-skills מהשנה שעברה היו הרבה implementation details, כמו פקודות סקריפט מדויקות. עם המודלים החדשים אפשר למחוק אותם ולהשאיר רצף צעדים של תהליך, ולכן "I think over time we'll see that skills get smaller and smaller" [1:00:53].

## 9. מה לעשות מחר בבוקר

- **לבנות verification skill לאפליקציה המרכזית:** שה-agent יוכל להריץ, ללחוץ, לקחת trace ולראות תוצאה. זה התנאי לכל השאר.
- **להוציא את החלק הדטרמיניסטי מה-skills:** לעבור על כל skill ו-rule ולשאול מה מהם מכני. את החלקים האלה להעביר ל-CLI או לסקריפט בתוך ה-skill.
- **להפוך כל טעות חוזרת ל-lint rule: constraint:** טיפוס או convention של מבנה תיקיות. לפרק god files לתיקיות לפי feature.
- **לזכור transcripts:** לבקש מה-agent למצוא בשיחות הקודמות את ההתערבויות החוזרות שלך ולהציע skills או lint rules.
- **לחבר outer loop אחד:** ערוץ באגים ב-Slack או ב-Linear שמגיע ל-agent, שמשחרר את הבאג על main לפני תיקון.
- **להשתמש ב-buffer לממצאי סריקה:** לרשום ממצאים למסמך ולסקור אחת לכמה ימים, במקום להפעיל agent לכל ממצא.
- **לעבור לדגימה רק אחרי שה-verification הוכח:** ולהתחיל במשימות שהן two-way doors.

| שאלה פתוחה                                                       | למה היא פתוחה                                     |
|------------------------------------------------------------------|---------------------------------------------------|
| מה שיעור ה-revert והרגרסיות ב-PRs שמוזגו אוטומטית?               | לא נמסר אף מספר על איכות.                         |
| כמה עולה full autopilot בטוקנים, ל-PR ולחודש?                    | נאמר רק [50:06] quite token intensive.            |
| איך עושים את זה בתחומים שבהם רוב ה-PRs הם one-way doors?         | היא אמרה במפורש שאין לה תשובה [54:42].            |
| איך הצוות האנושי צורך 2,500 PRs בחודש?                           | השאלה של Matt נענתה ב-sampling ולא בחוויית הצוות. |
| מה השם המדויק של GrokBot, והאם הוא זמין מחוץ ל-Cursor/SpaceX AI? | ה-ASR נותן ארבעה איתנים שונים, ולא הוצג מסך.      |

## 10. מילון מונחים

| מונח                               | משמעות                                                                                           |
|------------------------------------|--------------------------------------------------------------------------------------------------|
| Trust ladder                       | סולם האמון: ככל שסומכים יותר על ה-agent אפשר לתת לו משימות גדולות יותר ולהריץ יותר agents במקביל |
| Verification                       | היכולת של ה-agent להריץ את הקוד, לתקשר איתו כמו משתמש ולבדוק את העבודה של עצמו                   |
| Hill climbing                      | שיפור איטרטיבי מול rubric או ציון, בלולאה שה-agent מריץ לבד                                      |
| Inner loop / Outer loop            | לולאה פנימית: agents שעובדים על הקוד. לולאה חיצונית: context שמגיע מ-Slack, Linear, X ומייל      |
| Coordinator agent / Chief of staff | agent מנהל שלא כותב קוד בעצמו, מחלק משימות ל--sub agents ורואה את התמונה הכוללת                  |
| Gardening                          | תחזוקה ושיפור שוטפים של הקוד והסביבה, בניגוד לפיתוח features                                     |
| Dark factory                       | מפעל שרץ בלי בני אדם. כאן: agents שממזגים קוד בלילה בלי review מקדים                             |
| God file                           | קובץ ענק שמרכז הרבה אחריות (כאן: לפחות 10,000 שורות)                                             |
| Type narrowing                     | צמצום טיפוס רחב לטיפוס ספציפי דרך type guards ובדיקות                                            |
| One-way / Two-way door             | החלטה שאי אפשר לבטל, מול החלטה שאפשר להחזיר אחורה בזול (revert)                                  |
| Fuzzing                            | הפעלה אקראית של האפליקציה כדי למצוא באגים ורגרסיות                                               |
| Codemod / AST                      | טרנספורמציה מכנית של קוד על בסיס עץ התחביר, בלי שיקול דעת של agent                               |
| Tautological tests                 | טסטים שבודקים את עצמם ולא מוכיחים דבר                                                            |
| Compaction                         | סיכום אוטומטי של ה-context כשחלון ה-context מתמלא                                                |