כל קשת בגרף מכילה:

* מי הצמתים שביניהם היא מחברת
* התווית שלה
* רמת הידע שיש לנו עליה- איבר ב-lattice

כל צומת בגרף מכיל:

* אינדקס
* אינדקס של קבוע שהאובייקט שווה אליו
* קשת ל"אב ביולוגי" (אם קיימת)
* רשימת שכנים (מופרד לאבות ולבנים). לכל תווית יכול להיות לכל היותר בן אחד, אבל הרבה הורים.
* רמת הידע על הצומת- איבר ב-lattice, שבפועל יהיה L\_TOP או L\_MUST\_HAVE

הגרף:

* הגרף הוא DAG (עד כדי לולאות עצמיות)
* כל קודקוד הוא או TOP, או שיש לו אב ביולוגי, או שיש לו קבוע

פונקציות עיקריות בגרף:

* unlink\_vertex

המצב האבסטרקטי:

* מכיל מיפוי ממשתנים ראשיים (ללא attributes) לצמתים בגרף
* תומך ב:
  + query- האם קיים משתנה מסוים
  + set\_var\_to\_var- עדכון כאשר נעשית השמה מהצורה x=y, כאשר y אינו משתנה פשוט
  + set\_var\_to\_const- עדכון כאשר נעשית השמה מהצורה x=y, כאשר y הוא משתנה פשוט.
  + clone- החזרת מצב אבסטרקטי חדש וזהה לקיים
  + lub- ביצוע פעולת lub בין שני מצבים אבסטרקטים.

Lub בין מצבים אבסטרקטים

בשלב ראשון, נשנה את שמות הצמתים והקבועים, כך שלא יהיו שמות זהים (פרט לשורש שיישאר 0 בשני הגרפים).

נעבור על שני הגרפים (ב-DFS), ונחלק את הצמתים והקשתות ש:

* נמצאים רק בגרף הראשון
* נמצאים רק בגרף השני
* נמצאים בשני הגרפים יחד

כאשר הצמתים שמתאימים למשתנים הראשיים הם נקודות ההתחלה של שוויונות בין צמתים.

טיפול ב:

* קשתות משותפות:
  + עושים lub בין ה-knowledge שלהן.
* צמתים משותפים:
  + כדגכגדכ
* צמתים וקשתות לא משותפים:
  + כל הקשתות הופכות להיות L\_MAY\_HAVE (כולל הקשת שמחברת צמתים "ראשיים" לצומת -1)
  + צמתים לא נהפכים להיות TOP.