

BTP Phase 1

by MAYANK WADHWANI

Submission date: 20-Nov-2020 10:11PM (UTC+0530)

Submission ID: 1451615130

File name: Phase_1_BTP_Report.pdf (426.22K)

Word count: 3948

Character count: 18039

Admissible Strategies in Safety Games

Aman Raj - 170101006

aman170101006@iitg.ac.in

Indian Institute of Technology, Guwahati

ABSTRACT

Controller Synthesis revolves around the design and implementation of systems working in an environment to satisfy their goals. A popular approach to solve controller synthesis is to find winning strategies in graph games for the system against the environment. **Safety games** are a class of graph games in which the winning objective for a player is to never visit their corresponding unsafe states. However, in several cases where no winning strategies exist, we find **admissible strategies** in which players play rationally instead of adversarially and achieve their goals. In our work, we implement algorithms to find winning and admissible strategies for safety objectives in graph games.

KEYWORDS

Graph Games, Safety Games, Admissible Strategies

1 INTRODUCTION

Games played on graphs with finite vertices have been a topic of study for various years with applications lying in a plethora of examples like controller synthesis. Given, a model of the assumed behaviour of the environment and a system goal, controller synthesis aims at producing a behavioural model for a component that when executing in an environment consistent with the assumptions results in a system that is guaranteed to satisfy the goal. An approach to solve this controller synthesis problem involves graph games.

The **controller synthesis** problem for reactive systems can be modelled into an interactive game of strategy typically between two players, *Player 0* being the system and *Player 1* being its environment. These so-called **Graph Games** are played infinitely on a finite graph since there are no dead ends. Vertices are partitioned between players and the player owning the vertex decide the next move from that vertex.

There are different winning conditions depending on the required specification of the controller. It could be reachability, safety, etc. A strategy followed by a player is the mapping of moves the player makes on each of its vertex. If a strategy allows the *Player 0 (the system)* to satisfy the required specification or achieve his goals, no matter what strategy

Mayank Wadhwani - 170101038

mayan170101038@iitg.ac.in

Indian Institute of Technology, Guwahati

Missing "," ETS

the opponent *Player 1 (the environment)* follows, then it is called a **winning strategy** for *Player 0*.

Safety Games are a class of graph games which involves safety objectives. The condition in safety games is that *Player 0* is not allowed to enter a set of unsafe/bad vertices, i.e., the whole game play should be confined to a set of safe vertices. To find a winning strategy for *Player 0* in safety games, we make use of its duality with Reachability games, convert the safety objective into reachability objective and implement the algorithm used to find the winning strategy for reachability games.

But, there exists many scenarios of safety games where no winning strategy exists for *Player 0*, in such cases, we look for best possible strategies which are not dominated by the opponent. These strategies are called **admissible strategies** or **non-dominated strategies**. Generally, to find a winning strategy, both players play in an adversarial manner, but admissible strategies include those ones in which *Player 1* plays **rationally** or co-operates and focuses on achieving its goal and if possible, without dominating *Player 0*. This way, we can find strategies which could lead to a win-win situation for both the players. In other words, *Player 1* helps *Player 0* win as long as he is winning.

One such problem that lies in this domain is the robot motion planning problem which is the subject of our Bachelor's Thesis Project. We will be working on finding admissible strategies for robot motion planning using various algorithms as described in [1] and [2]. Robot Motion Planning is a multi-agent game where each robot seeks to satisfy its goal. The robots may co-operate with each other in this process. So finding admissible strategies for them appears to be ideal.

This paper covers the process of **iterated elimination of dominant strategies** which helps us obtain **admissible strategies** for the safety games with no winning strategy.

2 PRELIMINARIES

The following section contains a background of the different terminologies that will be used in the later sections. We will be referring to the game described in Fig 2.

2.1 Arena

An arena is defined as a tuple $A = (V, V_0, V_1, E)$.

Here, V represents the total set of all vertices in the graph.

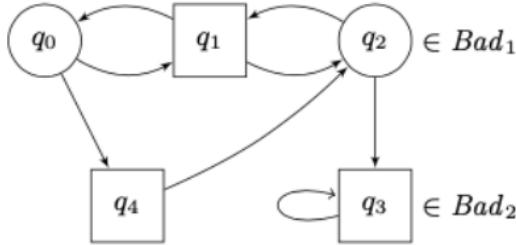


Figure 1: Example of a Game

V_0 represents the set of vertices for the first player (as denoted by Player 0),

V_1 denotes the set of vertices for the second player and finally E denotes the set of edges between 2 vertices of the graph.

Informally, an arena is a directed graph which contains edges between vertices representing the different players playing the game. In the example, the q_0 and q_2 will belong to the set V_0 and the remaining would belong to the set V_1 .

2.2 Controlled Predecessor

Given a player i and some set R , the controlled predecessor $CPre_i(R)$ of R is defined as:

$$\begin{aligned} CPre_i(R) &= \{v \in V_{i-1} \mid \forall \alpha \in R, (v, \alpha) \in E\} \cup \\ &\quad \{v \in V_i \mid \exists \alpha \in R, (v, \alpha) \in E\} \end{aligned}$$

Informally, the controlled predecessor returns the set of all vertices from which if we start, we are bound to reach some vertex of the given set R .

So we break this problem into two, finding all vertices that belong to same set of the player and check if there is any vertex from which we can reach R . If yes, we simply add the vertex to the set.

Now, if the vertex doesn't belong to the player's vertices, then we check if all the outgoing edges from this vertex would eventually lead to R . If it does, the second player will have no option but to visit the R .

2.3 Attractor

Given, a Player i and some set R for a reachability game, the i -attractor given by $Attr_i(R)$ of set R in arena A is defined inductively using the controlled predecessors defined in section 2.2 as follows:

- $Attr_i^0(R) = R$
- $Attr_i^{n+1}(R) = Attr_i^n(R) \cup CPre_i(Attr_i^n(R))$
- $Attr_i(R) = \bigcup_{n \in \mathbb{N}} Attr_i^n(R)$

While running this process, after some iterations, no new vertex gets added in the Attractor. Generally, after at most $|V|$ steps, the stages of attractor converge and become stationary.

Article Error (ETS)

Winning region for Player i refers to the set of vertices of the arena from where Player i has a winning strategy. Hence, to get $Attr_i(R)$ which gives the winning region for the Player i denoted by $W_i(G)$, we need to compute only $Attr_i^{|V|}(R)$.

Proper Noun (ETS)

8

2.4 Multiplayer Games

A multiplayer game G is defined as a tuple:

$$G := \langle P, A, (Win_i)_{i \in P} \rangle$$

Sp. (ETS)

Here, P refers to the non-empty set of players involved in the game. A represents the arena of the game as defined in section 2.1. Win_i denotes the winning condition for each player i in the set players P .

In this paper, we will be focussing on 2-player games. So, $P = \{0, 1\}$. We will be looking at the games from perspective of Player 0 (system). Depending on the objective of the game, the winning condition could be a reachability condition, safety condition, etc.

Missing "," (ETS)

2.5 Reachability Games

A class of multiplayer games with reachability as its winning condition. The reachability condition $REACH(R)$ is defined as follows:

$$REACH(R) := \{\rho \in V^\omega \mid Occ(\rho) \cap R \neq \emptyset\}$$

Here, R called the reachability set is a set of vertices such that $R \subseteq V$. V refers to the total vertices of the arena. V^ω denotes various plays or sequences of moves possible in the game. $Occ(\rho)$ denotes the vertices reached atleast once in the play.

The aim of Player 0 is to reach at least one vertex from a set of vertices R once. Hence, the set of vertices common between $Occ(\rho)$ and R should not be null/empty. Player 1 shows adversarial play and tries to avoid him from doing so.

2.6 Safety Games

Safety objective is another type of winning objective involved in multiplayer games. This class of games have a safety condition $SAFETY(S)$ defined as follows:

$$SAFETY(S) := \{\rho \in V^\omega \mid Occ(\rho) \subseteq S\}$$

Here, S is a set of safe vertices such that $S \subseteq V$. V refers to the total vertices of the arena. V^ω denotes various plays or sequences of moves possible in the game. $Occ(\rho)$ denotes the vertices reached at least once in the play.

The aim of Player 0 is to remain confined in S always. In other words, for Player 0 to achieve its objective, at no point in the play, a non-safe or bad vertex from the set $V \setminus S$ should be reached. Hence, the set of vertices in $Occ(\rho)$ should be a subset of safe vertices S . Player 1 shows adversarial play and tries to avoid him from doing so.

If we observe this game from perspective of Player 1, it can be seen as a reachability game where the objective of

Article Error (ETS)

Algorithm 1: Check if winning strategy

Input: Arena $A = (V, V_0, V_1, E)$, Initial State, Bad States

Output: True if winning strategy exists from given state

Find Dual of given Input Graph (Assign all vertices for player 1 to player 0 and vice versa);

attractor_i = Good_States (Initially the given good states act as the Attractor);

while Length of attractor_i changed from previous iteration **do**

 Find $CPre_i(\text{attractor}_i)$;

 Updated_Attractor_i = attractor_i \cap $CPre_i(\text{attractor}_i)$;

 attractor_i = Updated_Attractor_i;

end

if initial_state not in attractor_i **then**

return True;

else

return False;

end

Article Error (ETS)

Player 1 is to reach any vertex from the set $V \setminus S$. This gives us the sense of *duality* between these two types of games.

Here, the strategy σ' is said to be dominated in S as σ dominates it. And a strategy which is not dominated by any other strategy is an *admissible strategy* in S .

To obtain the set of admissible strategies, we iteratively eliminate the dominated strategies.

9

2.7 Strategy

A strategy for a Player i in a game given by the arena $A = (V, V_0, V_1, E)$ is a function $\sigma: V \setminus V_0 \rightarrow V$, such that $\sigma(v) = v'$ where $v \in V \setminus V_0$ and $v' \in V$, and $(v, v') \in E$. Hence, σ represents the mapping of states to moves chosen by the Player i against Player $1-i$.

Proper Noun (ETS)

2.8 Winning Strategy

Given, a game $G = (P, A, (Win_i)_{i \in P})$, a strategy σ is called a winning strategy for the player i if from a vertex $v \in V$ if every play starting from vertex v following the strategy σ satisfies the winning condition Win_i for that player, irrespective of what strategy is followed by the opponent.

For safety games, a winning strategy σ for Player 0 from vertex v makes sure that if a play starts from vertex v following that strategy, no unsafe vertex is reached during the play irrespective of what moves are chosen by Player 1.

P/V (ETS)

2.9 Dominance for strategies

Given, a rectangular set of strategy profiles $S = \prod_{i \in P} S_i$ where S_i represents a set of strategies of Player i . A strategy σ is said to very weakly dominate another strategy σ' w.r.t S , expressed as $\sigma \succ_S \sigma'$, if from all possible states s , the following condition satisfies:

$$\forall \tau \in S_{-i}, Win_i^s(\sigma', \tau) \Rightarrow Win_i^s(\sigma, \tau)$$

A strategy σ is said to weakly dominate another strategy σ' w.r.t S , expressed as $\sigma \succ_S \sigma'$, if the following conditions satisfies:

- $\sigma \succ_S \sigma'$
- $\neg(\sigma' \succ_S \sigma)$

2.10 Value

After n th step of elimination of dominated strategies, the value of history h for Player i is defined as:

- if \exists a winning strategy from $last(h)$, then $Val_i^n(h) = 1$.
- if \nexists winning strategy from $last(h)$ even if the other player helps, then $Val_i^n(h) \in E$.
- in all other cases $Val_i^n(h) = 0$.

Proper Noun (ETS)

Article Error (ETS)

P/V (ETS)

Confused (ETS)

Informally, the value of a state for a player denoted by val_i^n is 1 if there is a winning strategy for the player from that state. This means that even if the second player plays in an adversarial fashion, the player will still end up winning the game.

Moreover, the value of 0 means that the player will always lose from this state even if the second player plays in favour of the player. For all the remaining cases, we assign the value 0.

Sp. (ETS)

3 ALGORITHMS USED (METHODOLOGY)

In this section, we briefly discuss about the algorithms that were employed in our implementation.

3.1 Checking for Winning Strategy

We discuss the algorithm 1 in this subsection which will be used to check if a winning strategy exists from a given initial state for a safety game.

Algorithm 2: Compute Iteratively Admissible Strategies

Input: Arena $A = (V, V_0, V_1, E)$, The winning conditions of each player (in this case set of bad states for each player)

Win_i

Output: Set of admissible strategies for the players

while $\exists T \in \mathcal{T}$ and $Err(T) > 1$ do

for $s \in \text{Controlled-Frag.}$ do

if \exists winning strategy for player i from s in graph then

$Val_i^n = 1$ [Article Error (ETS) Proper Noun (C) Missing Verb (ETS)]

else if \exists winning strategy for player i from s even if the other player helps in graph then

$Val_i^n = -1$ [Prop-Confused (ETS)]

else [Confused (ETS)]

$Val_i^n = 0$ [Confused (ETS)]

for $y \in V$ do

$\{x \in V \mid \text{P-Predecessor}(x, y) \in E \wedge x \in V_i \wedge Val_i^n(x) > Val_i^n(y)\}$

Garb-Confused (ETS)

$n = n + 1$

graph = graph \ $T^n - 1$

We first convert our safety game into its corresponding **dual**, the reachability game, in the process, all vertices belonging to player 0 in the safety game will now belong to player 1 in the reachability game and similarly for the other player also. As discussed above, player 0 will win the safety game if its dual ie. player 1 wins the reachability game or in other words player 0 loses the reachability game. It should be noted that player 0 of the safety game is not the same as Cap. the player 0 of the reachability game.

Now, to find if the player 0 will win the safety game, player 0 should lose the reachability game or the initial state should not be a part of the final value of attractor_i. This is because the final value of attractor_i denotes the winning region for player 0, ie. the set of all vertices from which we will eventually reach some vertex \in good states and if the initial state is not a part of this winning region, it means player 0 will lose the reachability game which is desired.

We now focus our attention towards finding the final value of attractor_i or the winning region for a reachability game. We can do this by looping till there is no more changes in the attractor_i and at each iteration finding the controlled predecessor as described above. We take the union of the found controlled predecessor with our attractor_i and if there is no change break the loop.

In this manner, we are able to find if there exists a winning strategy for a player.

NOTE: The algorithm described in [1] also requires us to find if from a state we will always lose, that is even if the other players help us, we are still bound to lose.

We have in our implementation solved this by a simple idea. We convert all vertices to player 0, ie. we assume that there is no 'other player' and so if there is only one player, it

will not play adversarial to itself. A player will never want to defeat itself. So if a winning strategy exists for the player from a given state, this means that there are still chances of the player to win the game but if no winning strategy exists for this case, we declare that the player will lose the game no matter how optimally it plays.

3.2 Computing Admissible Strategies

In this section, we briefly discuss about the second half of our algorithm where we find the admissible strategies for a given safety game. We do this by taking references from [1]. We initialize the T_i set to \emptyset and loop till we find ourselves in a condition where for all states of the graphs, the value of the set T did not change for an iteration, that is the set T has converged.

In a particular iteration, we iterate over all the vertices of the graph and compute the values of the state for the player i . We make use of the algorithms as discussed in the previous section. So we can compute whether or not a winning strategy exists or it never exists which corresponds to the values 1 and -1 respectively. If both of these are false, we simply assign the value 0 and continue.

We then for all players, iteratively construct the **T set**, by adding all those edges to this set where the value of start vertex is greater than the value of successor, ie. all those edges where we will move from a higher value to a lower value. We then remove all the edges present in this **T set** from the graph as we will never take those paths.

We continue this process till the above mentioned condition is reached and then we break the loop. In the end, we have the set of all admissible strategies for the players.

4 EFFORTS

Since we were new to the domain of controller synthesis, we had to initially spend a good amount of time in getting familiar with the different concepts in the field.

We read a couple of different research papers and lecture notes as provided by our mentor, Prof. Purandar Bhaduri. To maximize our learnings and to get a greater in-depth knowledge of the subject, we decided to **implement** the algorithms on our own from **scratch** for finding admissible strategies as presented in the paper by Brenguier et al.[1].

We implemented the algorithm as given in Zimmermann's lecture notes [2] to **find** if there exists a winning strategy for a player in a given safety game. We have tested the working of our implementation of different inputs as presented in the following sections and got expected **outputs**.

5 RESULTS

In this section, we present the **results** that **were** observed when we run our algorithm on different types of graphs as inputs.

5.1 Example 1

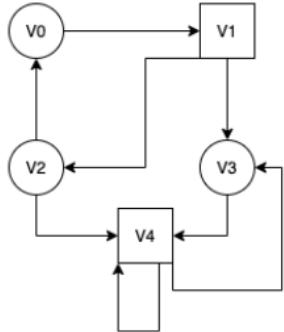


Figure 2: Example 1

In this example we have $v_3 \in \text{Bad}_0$ and $v_4 \in \text{Bad}_1$ or visiting v_3 will defeat player 0 and visiting v_4 will defeat player 1.

It can be seen that there **does not** exist any winning strategies for the players. So we find the admissible strategies using our algorithm. Our implementation is such that we print the set of edges that is to be deleted from the total set of edges. In other words, we print the edges that we would never visit or the final T set as defined above.

It was expected that we should remove the edge $v_2 -> v_4$. This is because if the player 0 plays this move and reaches v_4 , player 1 will lose the game, and player 1 from this point can act adversarial and move to v_3 which will defeat player 0 also. So player 0 instead will prefer moving to the vertex v_0 .

We also expect to remove the edge $v_1 -> v_3$ since again, if player 1 moves to v_3 and defeats v_3 , player 0 can move to v_4 to defeat player 1.

So we expected to remove the edges $v_1 -> v_3$ and $v_2 -> v_4$ and our actual results were in **accordance** with the expectations as can be seen from Fig 3.

```

[users-MacBook-Air:src mayankwadhwani$ python3 *.py us
1 -> 3
2 -> 4
users-MacBook-Air:src mayankwadhwani$ 
  
```

Figure 3: **Result** for Example 1
Article Error

5.2 Example 2

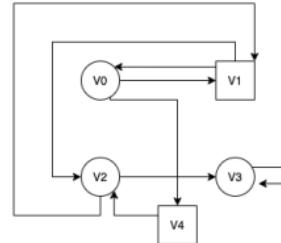


Figure 4: Example 2

In this example we have $v_2 \in \text{Bad}_0$ and $v_3 \in \text{Bad}_1$ i.e., visiting v_2 will defeat player 0 and visiting v_3 will defeat player 1.

Again, in this example also, it can be verified that player 0 does not have a winning strategy. We then try to compute the admissible strategies for the same.

It can be noticed that the **value** of q_4 initially for player 1 is -1 since if it reaches q_2 and defeats player 0, player 0 can act **adversarial** and defeat it. But its **value** from v_0 is 0, since if player 0 moves to q_1 , we can move back to q_0 which will form a cycle of length 2. So since **value** of q_0 is greater than the **value** of q_4 for player 1, we are expected to remove this edge.

In a similar fashion, one can argue to expect $v_1 -> v_2$ getting removed since if player 1 moves to q_2 and defeats player 0, it can move to q_3 and defeat player 1.

So we expected to see two edges $v_0 -> v_4$ and $v_1 -> v_2$ getting removed. This is what was observed after running our implementation.

We can therefore conclude that our **implementation** of finding admissible strategies for given safety games **yields corrects outputs**.

Wrong Form

Admissible Strategies in Safety Games,

```
[users-MacBook-Air:src mayankwadhani$ python3 *.py
0 -> 4
1 -> 2
users-MacBook-Air:src mayankwadhani$ ]
```

[3] Rajeev Alur, Salar Moarref and Ufuk Topcu. *Compositional and symbolic synthesis of reactive controllers for multi-agent systems*. August 2018, *rror* *Frag.*

Figure 5: Result

Article Error

6 PLAN OF ACTION AND FUTURE WORK

In this semester, we were able to familiarize ourselves with all the necessary concepts and were able to implement algorithms to find admissible strategies for giving safety games.

For the next semester, we will dive deep into the subject and start working on the **Robot Motion Planning** as given in Alur et al. [3]. More formally, we will be working on dynamically decoupled systems and will try to find admissible strategies if possible.

Dynamically decoupled systems are those systems where the agents (or the players) do not interact with each other, ie. the transition taken by a player from a state (also called as the dynamics) is not dependent on the other player. One such example of such a system is a robot motion planning system. We also have a notion of uncontrolled and controlled agents, where the controlled agents play in a co-operative way whereas the uncontrolled ones can play in an adversarial fashion.

Informally, we will be given 2 robots on an $n \times n$ grid, which will also have some obstacles where we are not allowed to go. Further we know that R2 is uncontrolled and R1 is controlled. Also R1 has imperfect sensors which will tell if there is any robot in the vicinity of distance 1, ie. in all the adjacent cells or corners.

We will first define the robot motion problem in an **LTL** (**Linear Temporal Logic**). Then we will be using methods as discussed in [3] to convert the LTL to a **safety game**. This is a fairly complex task. Once we get the corresponding safety game, we will be making use of the work done in this semester to find the admissible strategies. We will have to think of ways to use these strategies in our problem.

ACKNOWLEDGEMENTS

We would like to express our gratitude to our mentor **Prof. Purandar Bhaduri** for giving us the opportunity to explore this new field of Infinite Games and the much needed zest to delve into some of the state-of-the-art works.

REFERENCES

- [1] Romain Brenguier, Jean-Francois Raskin and Mathieu Sassolas. *The Complexity of Admissibility in Omega-Regular Games*. July 2014. <https://dl.acm.org/doi/10.1145/2603088.2603143>
- [2] Martin Zimmermann, Felix Klein, and Alexander Weinert. *Infinite Games: Lecture Notes*. Saarland University. Summer Term 2016.

BTP Phase 1

ORIGINALITY REPORT



PRIMARY SOURCES

1	Lecture Notes in Computer Science, 2002. Publication	2 %
2	dx.doi.org Internet Source	1 %
3	Walukiewicz, I.. "Pushdown Processes: Games and Model-Checking", Information and Computation, 20010129 Publication	1 %
4	fsttcs2016.cmi.ac.in Internet Source	1 %
5	"Handbook of Model Checking", Springer Science and Business Media LLC, 2018 Publication	1 %
6	drops.dagstuhl.de Internet Source	1 %
7	export.arxiv.org Internet Source	<1 %
8	Marcin Jurdziński, Mike Paterson, Uri Zwick. "A deterministic subexponential algorithm for	<1 %

solving parity games", Proceedings of the
seventeenth annual ACM-SIAM symposium on
Discrete algorithm - SODA '06, 2006

Publication

9

C.S. Jutla. "Tree automata, mu-calculus and
determinacy", [1991] Proceedings 32nd Annual
Symposium of Foundations of Computer
Science, 1991

<1 %

Publication

10

Lecture Notes in Computer Science, 2015.

<1 %

Publication

11

Yajima, Y.. "Characterizations of normal covers
of rectangular products", Topology and its
Applications, 20061101

<1 %

Publication

12

Bruno Courcelle, Igor Walukiewicz. "Monadic
second-order logic, graph coverings and
unfoldings of transition systems", Annals of Pure
and Applied Logic, 1998

<1 %

Publication

Exclude quotes

Off

Exclude matches

Off

Exclude bibliography

On

BTP Phase 1

PAGE 1



Missing "," You may need to place a comma after this word.



Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.



Article Error You may need to use an article before this word. Consider using the article **the**.



P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes the meaning unclear.



Possessive You may need to use an apostrophe to show possession.



Article Error You may need to remove this article.



Missing "," You may need to place a comma after this word.



Missing "," You may need to place a comma after this word.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

PAGE 2



Article Error You may need to use an article before this word.



Proper Noun If this word is a proper noun, you need to capitalize it.



Proper Noun If this word is a proper noun, you need to capitalize it.



Proper Noun If this word is a proper noun, you need to capitalize it.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Article Error You may need to remove this article.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes meaning unclear.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Missing ", " You may need to place a comma after this word.



Article Error You may need to remove this article.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes meaning unclear.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes meaning unclear.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Article Error You may need to use an article before this word. Consider using the article **the**.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Proper Noun If this word is a proper noun, you need to capitalize it.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes the meaning unclear.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Prep. You may be using the wrong preposition.



Confused You have used **A** in this sentence. You may need to use **an** instead.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes meaning unclear.



Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes meaning unclear.



Missing "," You may need to place a comma after this word.



Article Error You may need to use an article before this word.



Article Error You may need to use an article before this word. Consider using the article **the**.



Article Error You may need to use an article before this word.

PAGE 3



Article Error You may need to use an article before this word.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Article Error You may need to remove this article.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Prep. You may be using the wrong preposition.



Confused You have used **ð** in this sentence. You may need to use **an** instead.



Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.



Article Error You may need to remove this article.



P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize the sentence, you may want to revise it using the active voice.



Proper Noun If this word is a proper noun, you need to capitalize it.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Article Error You may need to remove this article.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Proper Noun If this word is a proper noun, you need to capitalize it.



Proper Noun If this word is a proper noun, you need to capitalize it.



Confused You have a spelling mistake near the word **ð'** that makes **ð'** appear to be a confused-v error.



Article Error You may need to use an article before this word.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Article Error You may need to use an article before this word.



Confused You have a spelling mistake near the word **ð'** that makes **ð'** appear to be a confused-v error.



Sentence Cap. Remember to capitalize the first word of each sentence.



Proper Noun If this word is a proper noun, you need to capitalize it.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have a spelling mistake near the word **ð'** that makes **ð'** appear to be a confused-v error.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize the sentence, you may want to revise it using the active voice.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize the sentence, you may want to revise it using the active voice.



Confused You have used **lose** in this sentence. You may need to use **loose** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Dup. You have typed two **identical words** in a row. You may need to delete one of them.



Confused You have a spelling mistake near the word **ð'** that makes **ð'** appear to be a confused-v error.

PAGE 4



Article Error You may need to use an article before this word. Consider using the article **a**.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Article Error You may need to use an article before this word.



Proper Noun If this word is a proper noun, you need to capitalize it.



Article Error You may need to use an article before this word.



Missing "," You may need to place a comma after this word.



Proper Noun If this word is a proper noun, you need to capitalize it.



Confused You have used **from** in this sentence. You may need to use **form** instead.



Confused You have a spelling mistake near the word **ð'** that makes **ð'** appear to be a confused-w error.



Confused You have a spelling mistake near the word **ð'** that makes **ð'** appear to be a confused-w error.



Proper Noun If this word is a proper noun, you need to capitalize it.



Proofread This part of the sentence contains a grammatical error or misspelled word that makes **ð'** meaning unclear.



Garbled Grammatical or spelling errors make the meaning of this sentence unclear. Proofread the sentence to correct the mistakes.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Confused You have used **ð'** in this sentence. You may need to use **an** instead.



Sentence Cap. Remember to capitalize the first word of each sentence.



Confused You have used **its** in this sentence. You may need to use **it's** instead.



Possessive You may need to use an apostrophe to show possession.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.



Sentence Cap. Remember to capitalize the first word of each sentence.



Confused You have used **find** in this sentence. You may need to use **fined** instead.



Prep. You may be using the wrong preposition.



Article Error You may need to remove this article.



Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.



Article Error You may need to remove this article.



Sentence Cap. Remember to capitalize the first word of each sentence.



Prep. You may be using the wrong preposition.



Article Error You may need to remove this article.



Article Error You may need to use an article before this word. Consider using the article **the**.



Article Error You may need to use an article before this word.



Sentence Cap. Remember to capitalize the first word of each sentence.



S/V This subject and verb may not agree. Proofread the sentence to make sure the subject agrees v the verb.



Hyph. You may need to add a hyphen between these two words.

PAGE 5



Wrong Form You may have used the wrong form of this word.



Confused You have used **from** in this sentence. You may need to use **form** instead.



Confused You have used **find** in this sentence. You may need to use **fined** instead.



Article Error You may need to use an article before this word.



Wrong Form You may have used the wrong form of this word.



S/V This subject and verb may not agree. Proofread the sentence to make sure the subject agrees with the verb.



Article Error You may need to use an article before this word.



Confused You have a spelling mistake near the word **its** that makes **its** appear to be a confused-error.



Article Error You may need to use an article before this word. Consider using the article **the**.



Article Error You may need to remove this article.



Wrong Form You may have used the wrong form of this word.

PAGE 6



Article Error You may need to use an article before this word.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Article Error You may need to use an article before this word. Consider using the article **a**.



Article Error You may need to remove this article.



Sentence Cap. Remember to capitalize the first word of each sentence.



Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.



Sentence Cap. Remember to capitalize the first word of each sentence.



Article Error You may need to remove this article.



Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.